

TECHNICAL WHITE PAPER

Highly Available Jenkins at Scale with Portworx Enterprise and AWS EKS

Gain an enterprise-grade CI/CD platform in the cloud.

Contents

About Jenkins	3
Introduction	3
Reference Architecture Diagram	4
Prerequisites	4
AWS EC2 Instance Sizing	
Utilities	
Configuring AWS EKS and Portworx	5
Installing and Configuring Portworx on Amazon EKS	9
Install the Portworx Operator	
Prepare Cluster for Installing Jenkins	14
Controller StorageClass	
Agent StorageClass	
AutoPilot Rules (Optional)	
Recommended AutoPilot Rule	17
Deploy Jenkins to Your Portworx-enabled EKS Cluster	
Deploy and Configure a Disaster Recovery Site	
Prepare Your AWS Cloud for Disaster Recovery	
Create a Cluster Pair and Admin Namespace	
Install and Configure PX-Backup	
Installation	
Configure PX-Backup	
Configure Your First Backup Job	
Solution Testing and Validation	33
Functional Testing: Jenkins Controller HA Failover Test	
Jenkins HA Failover	
Portworx Backup and Recovery UI	
Backup and Restore Jenkins	
Validate DR Replication Failover Preparedness	
Conclusion	
Appendix: Additional Testing	

About Jenkins

Jenkins is an industry-standard tool used by the DevOps community to orchestrate and schedule software build processes and continuous integration and continuous delivery (CI/CD) pipelines. Jenkins' architecture of controller and agent nodes lends itself well to dynamic environments, like those available in the public cloud such as Amazon Web Services (AWS). With the accelerated adoption of containers and new automation and orchestration tools like Kubernetes, developers are turning more frequently to managed solutions like Amazon Elastic Kubernetes Service (Amazon EKS). This ability to automate software builds, testing, quality checks, and continuous deployment brings great flexibility to how Jenkins can be leveraged. The flexibility of the cloud-native technologies has brought dramatic improvements in "time to value," as well as providing both the development and deployment environments for the enterprise.

Introduction

Running production-grade and enterprise-ready Jenkins in the cloud requires an elastic, resilient, and fault-tolerant architecture. Jenkins agents are very memory- and IO-intensive workloads and the on-demand nature of the cloud is ideal for this. However, the need for high-performance access to data and automating processes around both provisioning and protecting that data has driven many organizations to choose between high availability (HA) or performance. Amazon Elastic Block Store (Amazon EBS)volumes are an ideal option to meet the performance requirements of a CI/CD pipeline, but they are bound by the borders of the availability zone (AZ) where they were created. Customers trying to implement HA workloads in the cloud can benefit from Portworx® solutions to enable cross-AZ replication, business continuity, and data protection with a cloud-native approach. Portworx by Pure Storage® provides an automation and orchestration layer for data management that allows enterprises to maintain the high availability they need while running Jenkins on EBS.

Portworx is a data management solution that serves applications and deployments in Kubernetes clusters. Portworx itself is deployed natively within Kubernetes and extends the automation capabilities down into the infrastructure to remove the complexities of managing data. Portworx provides simple and easy-to-consume storage classes that are usable by stateful applications in a Kubernetes cluster. In AWS, Portworx does this by claiming EBS volumes that are attached to the worker nodes of a Kubernetes cluster (EC2 instances). These volumes are then abstracted by the Portworx data management control plane to deliver a storage pool that offers and automatically provisions container granular volumes from these available resources. When an application like Jenkins creates a persistent volume claim (PVC) with a Portworx storage class, Portworx will automatically provision container volumes and address the capacity, level of performance, data protection, security, and availability required for the application. Portworx is topology-aware and ensures that as part of the provisioning process, a replica of the data is maintained on another node in another availability zone. In the event of a failure, Portworx influences the Kubernetes scheduler to restart Jenkins in another AZ alongside the replicated data in a matter of seconds. With Portworx, clusters can be *highly dense*, meaning that you can greatly scale the number of containers per host. While the

Data protection is critical for any IT endeavor, but traditional, virtual-machine-based data protection solutions have proven inadequate for providing recoverable backups of Kubernetes workloads. Built exclusively for containerized applications, Portworx PX-Backup solves these shortfalls and protects your applications, including data, application configuration, and Kubernetes objects. It does this with a single click at the Kubernetes pod, namespace, or cluster level. Enabling application-aware, zero-data-loss backup and fast recovery for even complex distributed applications, PX-Backup delivers true multicloud availability.

Reference Architecture Diagram



Below is the reference architecture for AWS EKS and Portworx.

Figure 1. Portworx reference architecture

Prerequisites

AWS EC2 Instance Sizing

The instances used with Portworx must be sized to handle both the storage operations as well as the applications you will be deploying to the cluster. For Portworx, we recommend four vCPUs, 4GB of RAM, and 10Gbit networking at a minimum. We recommend using these as a baseline, and then factor in your application needs. In this example, we chose M5.xlarge instances (four vCPUs, 16GB RAM, 10G network) for Portworx, Jenkins, and agent pods to use. Your workload needs may vary.

Utilities

To complete the steps outlined in this guide, you will need to install and configure the following utilities:

AWS CLI v2: This is the primary component for interacting with AWS Cloud resources from the command line, and it will be needed for additional utilities to handle authentication from your management workstation. Additional information and installation instructions are available at https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html.

EKSCTL: This utility was developed for managing EKS clusters, and it is used here to deploy EKS resources. To install the latest release of eksctl, follow the instructions at <u>https://eksctl.io/introduction/#installation</u>.

KUBECTL: This is the primary CLI utility for Kubernetes, and it will be used extensively in this guide. You can install it using the instructions here: <u>https://kubernetes.io/docs/tasks/tools/</u>.

HELM v3: Helm is a popular package manager and deployment tool for Kubernetes, allowing application developers to package applications in a templated format enabling consistent deployments with all the necessary components included. Helm v3 should be installed based on the instructions at https://helm.sh/docs/intro/install/.

Configuring AWS EKS and Portworx

Documentation on creating a Portworx-enabled EKS cluster can be found on the Portworx website at https://docs.portworx.com/portworx-enabled EKS cluster can be found on the Portworx website at https://docs.portworx.com/portworx-enabled EKS cluster can be found on the Portworx website at https://docs.portworx.com/portworx-install-with-kubernetes/cloud/aws/. Portworx can also be deployed easily via the AWS Marketplace located here: https://portworx.com/awsmarketplace.

The recommended method to deploy a Portworx-enabled EKS environment is to use the eksctl utility and provide a configuration file. The reference guide can be found here: <u>https://docs.portworx.com/portworx-install-with-kubernetes/cloud/aws/aws-eks/eksctl/eksctl-operator/</u>.

Depending on the performance needs and the number of simultaneous agent tasks an organization might decide to run, implementing Portworx on EKS provides the flexibility to grow your development environment as your needs change. This unique functionality allows optimal resource use and provides true flexibility in how the deployment operates and scales. The number of storage nodes is entirely configurable. It is defined in the eksctl configuration file and can be adjusted to suit your needs at the time of deployment or modified later as utilization might change.

Here is a link to the available deployment architectures to help guide your decision: <u>https://docs.portworx.com/cloud-</u>references/deployment-arch/.

The first step involves granting Portworx permissions to create and attach EBS volumes. It is a critical step. To keep this process simple, we suggest implementing this at the instance level with an identity and access management (IAM) policy attached through the eksctl configuration file.

In the example **config.yaml** file below, the name of the segment identifier (SID) is set to "EKSPortworxEC2mgmt". The Amazon resource name (ARN) is then included as part of the configuration for your node groups so the provisioned hosts can perform the required tasks.

The IAM policy can be created using the AWS CLI or IAM console. In the console, select the option to create a new IAM policy. Navigate to the JSON tab and define your policy. Here is an example IAM policy that includes all the needed permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": " EKSPortworxEC2mgmt",
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:ModifyVolume",
                "ec2:DetachVolume",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DeleteTags",
                "ec2:DeleteVolume",
                "ec2:DescribeTags",
                "ec2:DescribeVolumeAttribute",
                "ec2:DescribeVolumesModifications",
                "ec2:DescribeVolumeStatus",
                "ec2:DescribeVolumes",
                "ec2:DescribeInstances",
                "autoscaling:DescribeAutoScalingGroups"
            ],
            "Resource": [
                "*"
            1
        }
    ]
}
```

If you are also deploying Portworx PX-Backup, then also create the following IAM policy using the JSON tab. This policy is named "EKS_PXBackup_Permissions" for this example.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EKS_PXBackup_Permissions",
            "Effect": "Allow",
            "Action": [
            "ec2:DeleteSnapshot",
            "ec2:DescribeInstances",
            "ec2:CreateTags",
            "Ettime [
            "Ettime [
            "ec2:CreateTags",
            "Ettime [
            "Ettime [
            "ec2:CreateTags",
            "Ettime [
            "Ettim
```

TECHNICAL WHITE PAPER

```
"ec2:CreateSnapshots",
    "ec2:DescribeVolumes",
    "ec2:CreateSnapshot",
    "ec2:DescribeRegions",
    "ec2:DescribeSnapshots",
    "ec2:CreateVolume"
    ],
    "Resource": "*"
  }
]
```

Once the policies have been created, create your cluster. Note that the resulting IAM policies are attached to the node group in the eksctl configuration file below.

Deploy EKS cluster using a variation of the following config.yaml file:

NOTE: Values in <brackets> should be changed to reflect your needs.

NOTE: For NodeGroup a minimum of three storage nodes are required. These can perform both workload and storage operations. More than three can be allocated based on a customer's need; however, it is recommended that the number of storage nodes per availability zone be changed in the Portworx spec generator or example file below to reflect any changes to the total number of storage nodes.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: <Cluster Name>
  region: <Region>
  version: 1.20
iam:
  withOIDC: true
addons:
- name: vpc-cni
  attachPolicyARNs:
  - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
managedNodeGroups:
  - name: storage-node
    instanceType: <m5.xlarge> # Select Instance type with minimum 4 vCPUs, 8Gi of Memory, and 10Gbit
networking for optimal performance
    minSize: 3 # Minimum configuration - Change to suite needs
    maxSize: 3 # Storage Nodes Min and Max must be set to equal values
    volumeSize: 30
    #ami: auto
```

iam:
withOIDC: true
addons:
- name: vpc-cni
attachPolicyARNs:
- arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
managedNodeGroups:
- name: storage-node
instanceType: <m5.xlarge> # Select Instance type with minimum 4 vCPUs, 8Gi of Memory, and 10Gbit</m5.xlarge>
networking for optimal performance
<pre>minSize: 3 # Minimum configuration - Change to suit needs</pre>
maxSize: 3 # Storage Nodes Min and Max must be set to equal values
volumeSize: 30
#ami: auto
amiFamily: AmazonLinux2
labels: {role: worker, "portworx.io/node-type": "storage", "px/metadata-node":"true"}
tags:
nodegroup-role: worker
ssh:
allow: true
Set to the path for your key file. See eksctl documentation for more details.
publicKeyPath: < \$HOME/.ssh/id-rsa.pub >
iam:
attachPolicyARNs:
- arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
- arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
- arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
- arn:aws:iam::aws:policy/ElasticLoadBalancingFullAccess
- arn:aws:iam:: <awsaccountnumber>:policy/EKSPortworxEC2mgmt</awsaccountnumber>
If you are also installing PX-Backup, also create the optional IAM Policy above and add it here
also.
- arn:aws:iam:: <awsaccountnumber>:policy/EKS_PXBackup_Permissions</awsaccountnumber>
withAddonPolicies:
imageBuilder: true
autoScaler: true
ebs: true
fsx: true
efs: true
albIngress: true
cloudWatch: true
availabilityZones: [' <region-az1>', '<region-az2', 'region-az3']<="" td=""></region-az2',></region-az1>

To deploy a new EKS cluster using the above configuration, save the file with the name cluster-config.yaml and issue the following command: eksctl create cluster -f cluster-config.yaml.

Allow up to 30 minutes after issuing the command to create your EKS cluster for it to become available in your AWS account. There are additional components that may be needed for a production-ready EKS cluster, so please consult Amazon's documentation for items like the ALB/ELB Load Balancer Controller, CertManager, and ExternalDNS for Route53.

Installing and Configuring Portworx on Amazon EKS

This section provides a walkthrough of the steps to install and configure Portworx on Amazon EKS; however, before proceeding, we recommend that you read the full documentation and instructions on the Portworx website https://docs.portworx.com/portworx-install-with-kubernetes/cloud/aws/aws-eks/. For additional information and options, the documentation website provides extensive resources for you to explore. Also, be sure to review all volume sizes and types in this document and adjust them to reflect your workloads. They are provided as examples and are likely to not align exactly with your needs.

Install the Portworx Operator

To install the Portworx Operator, run kubect1 create -f <u>https://install.portworx.com/2.7?comp=pxoperator</u>.

Once the operator is installed, we need to provide a specification file to deploy the Portworx cluster. Navigate to https://central.portworx.com to generate the needed Kubernetes manifest that you will deploy to your cluster. We have also included an example specification that you can deploy, but using the Portworx spec generator is preferred.

First, you will need to create an account on the Portworx website and log in. After logging in to the website, you are presented with the spec generator wizard. On the first tab, select **Portworx Enterprise** and click **> Next** in the bottom left corner.



On the next page , you'll find the basic configuration options.

On this page, make the following selections:

- 1. Select Use the Portworx Operator.
- 2. For Portworx Version, select the latest available version (presently 2.7).

	← Spec Generator - Enterpri	se		\$	
	Basic	Storage	Network	Customize	
			.	\$	
?	Use the Portworx Operator	0			
	Portworx Operator only supports ki	ubernetes versions 1.12 and up.			
	Portworx Version * 🔀 px-enterpris	e			
	2.6				
	ETCD * 🕄	Your etcd details 3	🧿 Built-in 🖲		
	Portworx will create and manage ar	internal key-value store (kvdb) cluster.			
	You can restrict the nodes that will run the key-value store by labelling your nodes with the label px/setadata-node=true. Only the nodes with the label will participate in the kvdb cluster. This allows you to use nodes with dedicated bardware for the key-value store.				
	For example: kubectl label nodes	nodel node2 node3 px/metadata-node=true			
	Pasat				
	Reset			Back Next	

Click **Next**, and you are presented with the storage options. The following steps are for an example configuration for AWS: make sure that the volume type and size are appropriate for your deployment:

- 1. Select Cloud, and then select AWS.
- 2. Configure storage devices:
 - a. Select Create Using a Spec.
 - b. Under Select EBS Volume Type, select GP2 or IO1.

NOTE: When selecting cloud volume configurations, consider your workloads and the characteristics of the available drive. GP2 provides a solution for balancing cost and performance. IO1 is recommended for production workloads that need a consistent input/output operations per second (IOPS) and throughput.

- c. Size: 500
- d. Max storage nodes per availability zone: 1
- e. Select Auto create journal device.

> ← 3	Spec Generator - Enterprise			<
	E Basic ✓ Kubernetes Version: Builtin etcd	Storage	Network	Customize
	- Select your environment *			
0		Select Clou	ıd Platform *	
	aws	Grage Charlester	Azure	vSphere
	AWS	Google cloud/GKE	AZURE	vSphere
	If you plan to use EC2 instance storage or pla Select type of disk *	en to manage EBS volumes your own way, sele	ct "Consume unused" or "Use Existing disks".	Use Exising Disks ()
	Select type of disk *	Create Using a Spec Size (GB) * 500	Consume Unused Consume Unused IOPS required from EBS volume * Not Applicable	Use Exising Disks Add/Delete Spec Object
	Max storage nodes per avaliibility zor 1	ie (Optional)		
	Auto create journal device			
	Reset			Back Next

3. Click Next to proceed to the network setup.

On this page, unless you have configured multiple network interfaces on your EKS hosts, all values can be left at the default values.

>	←Sp	ec Generator - Enterprise			¢
4		Basic 🗸	🛢 Storage 🗸	Network	Customize
		ubernetes Version: uiltIn etcd	Cluster Environment: AWS Volume, size GB , 1 max drives.	.	\$
୭		Interface(s)			
Ū				auto	
				auto	
		Advanced Settings			
	F	Reset			Back

Click Next to continue to the Customize page:

- 1. At the top of the page, select Amazon Elastic Container Service for Kubernetes (EKS).
- 2. For environment variables, add ENABLE_ASG_STORAGE_PARTITIONING = true if you are using Auto Scaling groups.
- **3.** Under Advanced Settings, select the following:
 - a. Select Enable Stork.
 - b. If needed based on your use case, select Enable CSI.

- c. Select Enable Monitoring.
- 4. In Cluster Name Prefix, enter an appropriate value, such as the cluster name. .
- 5. In Secret Store Type, you can select your Kubernetes Secrets, but Portworx can also leverage AWS's KMS service, Hashicorp Vault, or other standard KMS solutions.

Click Finish and agree to the licensing agreement to generate the manifests and spec files.

Customize			
	None 0		
	OpenShift 4+ 🚯		
	PKS (Pivotal Contain	ier Service) 🚯	
	Amazon Elastic Cont	ainer Service for	
	Kubernetes (EKS) 🚯		
Environment Variables			
Registry And Image Settings			
Security Settings			
Advanced Settings			
Customize			
Environment variables			
Env Variable (1) - Name Value 0	ENABLE_ASG_STORA	GE_PAR = true	* +
	List of environment variables	(name,value pairs) that will be exported to Portworx.	
	https://docs.portworx.com/	unc/options.html#env-variables has the env variables list that	Portworx supports.
Registry And Image Settings			
Security Settings			
Advanced Settings			
Customize			
Environment Variables			
Registry And Image Settings			
Security Settings			
Advanced Settings			
Enable Stork			
Enable CSI			
Enable Monitoring			
Cluster Name Prefix		Secrets Store Type Kubernetes	
		Kubernetes	

The next page provides the instructions to	o install Portworx into your EKS cluster.
--	---

Portworx Operator	
You have opted to use the Portworx Ope Please make sure to install the deploym	rator for deployment. ent spec mentioned below.
Install the Portworx Operator Deployme @ @ kubect1 apply -f 'https://ins	<u>nt Spec and wait for it to be operational.</u> tall.portworx.com/2.6?comp=pxoperator'
Websetl apply -f 'https://ins operator=true&mc=false&kbver=&b=tru 350f-499e-9ad8-1f3bb27069fb&eks=tru	tall.portworx.com/2.6? e&mz=1&s=&22type%3Dgp2%2Csize%3D500%22&j=auto&kd=type%3Dgp2%2Csize%3D150&c=px-cluster-b0b573e5- e&stork=true&mon=true&st=k8s&e=ENABLE_ASG_STORAGE_PARTITIONING%3Dtrue&promop=true'
Save Spec	
Comma separated Tags	

The first step on this page was covered previously, and it is not necessary to repeat the instructions.



Once the operator has been deployed, you can apply your StorageCluster specification by copying and pasting the command shown here on the page:

📙 🤨 kubectl apply -f 'https://install.portworx.com/2.6?
operator=true&mc=false&kbver=&b=true&mz=1&s=%22type%3Dgp2%2Csize%3D500%22&j=auto&kd=type%3Dgp2%2Csize%3D150&c=px-cluster-b0b573e5-
350f-499e-9ad8-1f3bb27069fb&eks=true&stork=true&mon=true&st=k8s&e=ENABLE_ASG_STORAGE_PARTITIONING%3Dtrue&promop=true'

Finally, you are given the option to save the spec files in PX Central for later reference or reuse by entering a name and any relevant tags, and then clicking **Save Spec**.

Save Spec	* Required	
Spec Name* 🚯		
Spec Tags* 🚯		
Comma separated Tags		
Back		Download Save Spec
Back		Download Save Spec

Below is an example specification for StorageCluster that you can use. Simply save it in a file named storagecluster.yaml and apply it to the cluster. The Portworx Operator will handle the rest of the deployment. It takes approximately 5-10 minutes for Portworx to fully initialize the cluster.

TECHNICAL WHITE PAPER

```
kind: StorageCluster
apiVersion: core.libopenstorage.org/v1
metadata:
name: px-cluster-jenkins
namespace: kube-system
annotations:
   portworx.io/is-eks: "true"
spec:
 image: portworx/oci-monitor:2.7.2
 imagePullPolicy: Always
 kvdb:
   internal: true
 cloudStorage:
   deviceSpecs:
   - type=gp2,size=500
 journalDeviceSpec: auto
 kvdbDeviceSpec: type=gp2,size=150
 maxStorageNodesPerZone: 3
 secretsProvider: k8s
 stork:
   enabled: true
   args:
     webhook-controller: "true"
 autopilot:
   enabled: true
   providers:
   - name: default
    type: prometheus
   params:
     url: http://px-prometheus:9090
 env:
 - name: "ENABLE_ASG_STORAGE_PARTITIONING"
 value: "true"
 monitoring:
   prometheus:
     enabled: true
     exportMetrics: true
```

Prepare Cluster for Installing Jenkins

To prepare the cluster to install Jenkins, first create a new namespace in your EKS cluster to run Jenkins: kubectl create namespace Jenkins.

Now that Portworx is installed and the StorageCluster object has been defined, create two Kubernetes StorageClass objects for Jenkins. The first will be used for the controller pods and the second for agent pods.

Controller StorageClass

The first step to create a StorageClass object for the controller pods is to create and save a file named controller-sc.yaml and apply it using kubectl. The contents of this file should be:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: px-sharedv4-sc
provisioner: kubernetes.io/portworx-volume
parameters:
    repl: "3"
    sharedv4: "true"
    sharedv4_mount_options: "vers=4.0"
    io_profile: db_remote
allowVolumeExpansion: "true"
```

Next, create a file named controller-pvc.yaml with the following contents:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
    name: jenkins-home-pvc
    namespace: jenkins
    labels:
        app: jenkins
annotations:
        volume.beta.kubernetes.io/storage-class: px-sharedv4-sc
spec:
        accessModes:
        - ReadWriteMany
        resources:
        requests:
        storage: 100Gi
```

This will create a ReadWriteMany volume, which can be used for both active and active-passive Jenkins controller deployments.

Agent StorageClass

To create a StorageClass object for agent pods, create and save a file named jenkins-agent-sc.yaml and apply it using kubectl. The contents of this file should be:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
```

```
name: jenkins-agent-sc
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "1"
  priority_io: high
allowVolumeExpansion: "true"
```

AutoPilot Rules (Optional)

To allow your EKS cluster to dynamically grow your persistent volume claims, you will need to define the AutoPilot rules. There are two rule types that you can define:

- Storage pool capacity rule: This rule allows Portworx to allocate additional EBS volumes automatically through monitoring and API calls. This rule is optional and should be used for workloads where there is not a clear understanding of the storage needs.¹
- **Persistent volume claim rule:** This type of rule allows the PVC for your Jenkins controller to dynamically grow, allowing you to start small and expand it within the bounds of your Storage Pool based on utilization or a user-defined limit. It is highly recommended that you use this rule for Jenkins deployments.

Please note the thresholds and capacity controls to ensure you are optimizing your pool of resources for your workload.

Below is an example of an optional AutoPilot storage pool rule.

```
apiVersion: autopilot.libopenstorage.org/v1alpha1
kind: AutopilotRule
metadata:
  name: pool-expand
spec:
  enforcement: required
 ###### conditions are the symptoms to evaluate. All conditions are AND'ed conditions:
  expressions:
  # pool available capacity less than 70%
  - key: "100 * (px_pool_stats_available_bytes / px_pool_stats_total_bytes)"
  operator: Lt
  values:
  - "70"
  # pool total capacity should not exceed 2TB
  - key: "px_pool_stats_total_bytes / (1024*1024*1024)"
 operator: Lt
 values:
 - "2000"
 ##### action to perform when condition is true
 actions:
```

```
<sup>1</sup>Pool-level capacity management with Autopilot requires additional licensing.
```



```
name: "openstorage.io.action.storagepool/expand"
params:
# resize pool by scalepercentage of current size
scalepercentage: "50"
# when scaling, add disks to the pool
scaletype: "add-disk"
```

Recommended AutoPilot Rule

The following AutoPilot rule for resizing the Jenkins controller volume PVC is recommended:

```
apiVersion: autopilot.libopenstorage.org/v1alpha1
kind: AutopilotRule
metadata:
  name: volume-resize
spec:
  ###### selector filters the objects affected by this rule given labels selector:
  matchLabels:
   app: jenkins
  ###### namespaceSelector selects the namespaces of the objects affected by this rule
  ###### conditions are the symptoms to evaluate. All conditions are AND'ed conditions:
  # volume usage should be less than 70%
  expressions:
  - key: "100 * (px_volume_usage_bytes / px_volume_capacity_bytes)"
  operator: Gt
  values:
  - "70"
  ##### action to perform when condition is true
  actions:
  - name: openstorage.io.action.volume/resize
   params:
  # resize volume by scalepercentage of current size
    scalepercentage: "50"
  # volume capacity should not exceed 400GiB
    maxsize: "400Gi"
```

Deploy Jenkins to Your Portworx-enabled EKS Cluster

At this point, you are now ready to install Jenkins. Jenkins is packaged in a Helm chart for easy deployment.

Next, add the Jenkins repository to your Helm installation with the following commands:

```
helm repo add jenkinsci https://charts.jenkins.io
helm repo update
```

Then you will need to obtain the values.yaml file from the Helm chart with the following command:

```
curl -o jenkins-values.yaml \ https://raw.githubusercontent.com/jenkinsci/helm-
charts/main/charts/jenkins/values.yaml
```

Review the contents of the values.yaml file for any customizations you may need. To complete the setup of Jenkins with Portworx, create an override-values.yaml file with the following contents. Other customizations may be needed to accommodate your CI/CD pipeline.

```
controller:
initializeOnce: true
schedulerName: "stork"
agent:
fsgroup: 1000
volumes:
 - type: PVC
claimName: jenkins-home-pvc
 mountPath: /var/jenkins_home
readOnly: false
 workspaceVolume:
 - type: DynamicPVC
   storageClassName: jenkins-agent-sc
  requestSize: 20
  accessModes: ReadWriteOnce
persistence:
 enabled: true
 existingClaim: jenkins-home-pvc
```

To deploy using these settings, issue the following command: helm install jenkins -f values.yaml -f overridevalues.yaml jenkinsci/jenkins -n jenkins

This will deploy Jenkins to your cluster with Portworx orchestrating your storage operations. Follow the instructions provided at the end of the install to obtain the URL and username/password to access your Jenkins environment.

IMPORTANT: Jenkins uses a Config Map to maintain the configuration during pod restarts. It is *highly* advised that you review the "Configuration as Code" section under "Managing Jenkins" in the Jenkins UI and use it as a reference to keep the Config Map consistent with your running configuration. If you chose not to do this, some configuration changes can be lost if the pod restarts.

Deploy and Configure a Disaster Recovery Site

Portworx provides a true disaster recovery (DR) and business continuity (BC) solution with the ability to replicate your applications between two different EKS clusters in different availability zones or regions, or other variations of Kubernetes that might live outside the Amazon ecosystem. Using this capability allows businesses to ensure that their most critical applications are always available.

Portworx Enterprise can provide both synchronous and asynchronous disaster recovery configurations, allowing you to build a DR/BC capability for almost any scenario. Metro-DR, or synchronous replication, requires a connection between the sites with adequate bandwidth and a maximum of 10ms latency between the EKS hosts. This is only achievable using a single region and multiple availability zones. This can offer cost savings, but it lacks true business continuity if the selected region suffers a catastrophic event.

For this white paper, region-to-region DR was selected as the preferred DR solution. This is configured as Async-DR using our 3D-Snapshot capabilities that include the Kubernetes objects and manifests that are needed, along with the data volumes to maintain an application-consistent DR site. Replication schedules are configurable by the cluster administrator and offer as low as a 10-minute recovery point objective (RPO) and the ability to restore service in minutes.

Prepare Your AWS Cloud for Disaster Recovery

Portworx Disaster Recovery–enabled clusters use an S3 bucket as an intermediary storage location that is accessible from both EKS clusters. As part of the configuration process, we use an AWS access key and a secret key. To ensure that you maintain a secure environment, it is best practice and strongly recommended that a separate IAM identity is created for this purpose. The identity should have the two IAM policies defined at the beginning of this guide attached to it, as well as policies allowing interaction with S3. Once you have created this identity and created access and secret keys, make sure to save them for future use in this white paper. When AWS account information is part of a step, please use this new IAM Identity to satisfy the requirements.

To configure Disaster Recovery, first select a second region and deploy a similar EKS cluster. The most important factor is that you need at least three storage nodes to accommodate the configuration used in the Jenkins home directory storage class. If look back, you will see that a replication factor of three is used to ensure the highest availability.

You can follow the above instructions to deploy a second cluster by changing the values necessary to reflect the use of a different region. Once that is done, it is necessary to create a VPC peering connection or transit gateway to allow connectivity between the clusters. See the links below to determine which will work best in your environment. Also, be sure to add rules to the security groups associated with both clusters to allow bidirectional communication. Once connectivity has been verified, you are ready to create a cluster pair and configure replication. For more information on VPC peering, see https://docs.aws.amazon.com/vpc/latest/peering/create-vpc-peering-connection.html. For more information on transit gateways, see https://docs.aws.amazon.com/vpc/latest/tgw/tgw-getting-started.html.

Create a Cluster Pair and Admin Namespace

To begin pairing the two clusters for replication, first establish an admin namespace. This is recommended so cluster administrators can replicate any namespace in the cluster without having to have full access to the kube-system namespace. Here are the instructions to configure and begin using an admin namespace with Portworx Enterprise:

- First, obtain the storkctl command-line utility from either cluster using the platform-specific instructions found on the Portworx documentation site. There are instructions for Linux, macOS, and Windows available. More information is available at: <u>https://docs.portworx.com/portworx-install-with-kubernetes/disaster-recovery/async-dr/</u>.
- Once you have the storkctl utility installed, you then need to provide credentials to Stork so it can access the needed resources for migrations. Use the IAM Identity's credentials in your AWS account, and ensure that this account has the same roles that were applied to the EKS clusters during deployment.

IMPORTANT: Until otherwise instructed, the following steps *must* be performed on both the source and destination clusters.

- 3. Create a Kubernetes namespace to serve as the admin namespace: kubectl create namespace migrations
- **4.** Next, create a secret in this new namespace using the AWS access key ID and secret access key created for the previously discussed IAM Identity. You should receive confirmation that the Kubernetes secret was created.

```
kubectl create secret generic --from-literal=aws_access_key_id=<AWS_ACCESS_KEY_ID> --from-
literal=aws_secret_access_key=<AWS-SECRET-ACCESS_KEY> -n kube-system aws_creds
```

- Edit the StorageCluster object in the kube-system namespace and add a volumeMount for the newly created secret. To
 edit the StorageCluster object, run kubect1 edit stc -n kube-system.
- 6. Move down through the manifest and find the stork: configuration and modify it to look like the following:

```
stork:
    args:
        admin-namespace: migrations
        health-monitor-interval: "30"
        webhook-controller: "true"
    enabled: true
    volumes:
    - mountPath: /root/.aws/
    name: aws-creds
    readOnly: true
    secret:
        secretName: aws-creds
```

Upon saving the changes to the StorageCluster object, a new replica set will be created and the Stork pods will restart to implement the changes in a rolling update.

7. On the destination cluster obtain the Cluster ID by issuing the following command:

```
PX_POD=$(kubectl get pods -1 name=portworx -n kube-system -o jsonpath='{.items[0].metadata.name}')
&& kubectl exec $PX_POD -n kube-system -- /opt/pwx/bin/pxctl status | grep UUID | awk '{print $3}'
```

Copy the output to a notepad-type app for use in the following steps. It will be referenced as <destination_cluster_uuid>.

To use the Portworx CLI, you can either log into one of your worker nodes and run the command from /opt/pwx/bin/ where it is located, or more conveniently create a local alias to access the CLI tool from inside a Portworx pod. For Linux or macOS, add the following to your shell configuration file (bash, zsh, etc.)

```
export PX_POD="$(kubectl get pods -1 name=portworx -n kube-system -o
jsonpath='{.items[0].metadata.name}')"
alias pxctl="kubectl exec $PX_POD -n kube-system -- /opt/pwx/bin/pxctl"
```

8. Now provide Portworx with the IAM Identity's AWS credentials. The command to use is:

```
pxctl credentials create --provider s3 --s3-access-key <AWS_ACCESS_KEY_ID> --s3-secret-key <AWS-
SECRET-ACCESS_KEY> --s3-region <region of source cluster> --s3-endpoint s3.amazonaws.com --s3-
storage-class STANDARD clusterPair_<destination_cluster_uuid>
```

As a reminder, all steps for creating a cluster pair to this point *must* be performed on both the source and destination clusters. The only exception is the command to obtain the Cluster ID.

IMPORTANT: The following steps will be performed on specific clusters.

9. To create the cluster pair, on the *destination* cluster, first, run this command and pipe it to a file for editing: storkctl generate clusterpair -n migrations remotecluster >> clusterpair.yaml.

Once the file has been created, edit the file and replace the contents of 'options' in the file with the following.

```
options:

ip: "<IP of any Portworx node in DR Cluster>"

port: "9001"

token: "<cluster token from DR Cluster>"

mode: DisasterRecovery<sup>2</sup>
```

10. To obtain the cluster token, run the following on the destination cluster: pxctl cluster token show. Use the output for the token requested above. Once the file has been modified, you will apply it to the source (production) cluster using: kubectl apply -f clusterpair.yaml

You can check the status of the cluster pair with: storkctl get clusterpair -n migrations. The output should look like this:

storkctl get	clusterpair -n mi	grations	
NAME	STORAGE-STATUS	SCHEDULER-STATUS	CREATED
remotecluster	Ready	Ready	08 Jun 21 17:44 EDT

Next, create a schedule policy for the replication. This is where you can configure how frequently the DR replication occurs. Currently, our shortest supported interval is every five minutes, resulting in an RPO of approximately 10 to 15 minutes.

Below is an example SchedulePolicy manifest that configures replication every 15 minutes, as well as a daily, weekly, and monthly example. Configure the replication interval based on your business needs and RPO policies. Multiple schedules can be created if different applications have different requirements.

```
apiVersion: stork.libopenstorage.org/v1alpha1
kind: SchedulePolicy
```

² Disaster recovery mode requires licensing additional features; please contact Portworx for more information. This line can be omitted, but every 7th replication interval will be a full copy versus forever incremental copies enabled by the DR license.

```
metadata:
    name: jenkins-dr-policy
    namespace: migrations
policy:
    interval:
    intervalMinutes: 15
    daily:
        time: "10:14PM"
    weekly:
        day: "Thursday"
        time: "10:13PM"
    monthly:
        date: 14
        time: "8:05PM"
```

Next, create a MigrationSchedule manifest that replicates the Jenkins namespace, where you have deployed Jenkins:

```
apiVersion: stork.libopenstorage.org/v1alpha1
kind: MigrationSchedule
metadata:
  name: jenkins-async-dr
  namespace: migrations
spec:
  template:
    spec:
      clusterPair: remotecluster
      includeResources: true
      includeVolumes: true
      startApplications: false
      namespaces:
      - jenkins
      adminClusterPair: remotecluster
      purgeDeletedResources: false
  schedulePolicyName: jenkins-dr-policy
```

Since you are configuring disaster recovery, take note that the options startApplications and purgeDeletedResources have been set to false. Migrations and migration schedules can be used for disaster recovery replication like you are doing here. But they can also be used for one-time application migrations between clusters or namespaces within a cluster by changing the kind to Migrations and omitting schedulePolicyName. This type of migration begins once the manifest is applied.

For disaster recovery, apply the manifest as displayed above to the source cluster; replication will begin shortly after. The first interval is a full copy of the selected namespaces. Every following replication interval uses incremental snapshots to minimize the amount of data sent. Also, on AWS, Portworx uses an S3 bucket to hold the migration data during the replication to help further reduce transport costs.

Each time the replication occurs, a migrations object is created in the migrations namespace. After 5-10 minutes, you can verify the status of the replication by issuing the following commands:

kubectl get migrations -n migrations
kubectl describe migrations <selected migration> -n kube-system

The output will list all objects that are part of the DR migration. For brevity, we have removed the bulk of the output; however, note that at the end of this block, "Stage: Final" has a status of successful.

```
# Some output was removed in the interest of brevity
Name:
              jenkins-async-dr-interval-2021-06-09-200912
Namespace:
              migrations
Labels:
              <none>
Annotations: <none>
API Version: stork.libopenstorage.org/v1alpha1
Kind:
              Migration
Status:
  Finish Timestamp: 2021-06-09T20:10:51Z
  Resources:
    Group:
                core
    Kind:
                PersistentVolume
    Name:
                pvc-2f05d999-b391-46cc-a4a2-bdee199c98e9
    Namespace:
    Reason:
                Resource migrated successfully
    Version:
                v1
    Group:
                rbac.authorization.k8s.io
    Kind:
                Role
    Name:
                jenkins-casc-reload
    Namespace: jenkins
    Reason:
                Resource migrated successfully
    Status:
                Successful
    Kind:
                RoleBinding
    Name:
                jenkins-watch-configmaps
    Namespace: jenkins
    Reason:
                Resource migrated successfully
    Status:
                Successful
    Version:
                v1
                Final
  Stage:
  Status:
                Successful
  Volumes:
    Namespace:
                              jenkins
    Persistent Volume Claim:
                              jenkins
    Reason:
                              Migration successful for volume
    Status:
                              Successful
                              pvc-2f05d999-b391-46cc-a4a2-bdee199c98e9
    Volume:
    Namespace:
                              jenkins
```

Persistent N	Volume	Claim:	jenkins-plugin-dir
Reason:			Migration successful for volume
Status:			Successful
Volume:			pvc-9a0d9983-ffab-4d7e-b11b-07cffb59c574
Namespace:			jenkins
Persistent N	Volume	Claim:	jenkins-plugins
Reason:			Migration successful for volume
Status:			Successful
Volume:			pvc-72e65c45-f916-4f83-9c9a-ead8f96f443f
Namespace:			jenkins
Persistent N	Volume	Claim:	sc-config-volume
Reason:			Migration successful for volume
Status:			Successful
Volume:			pvc-e4777535-2513-4cdf-98c5-8a1e9a3cb869

Let's look at a few important details. In the migration manifest, you specified that all objects from the jenkins namespace be replicated. This can be validated by looking at the output from querying the objects in the namespace on the target cluster.

Switching to that cluster, you can issue kubectl get all -n jenkins

kubectx px-jenkins-d Switched to context "px-	r1 -jenkins-dr1	1".					
<pre></pre>	<pre>>/px-jenking</pre>	s-eks/migrations	🛛 🖶 🖓 master	!2 ?5			px−jenkins-dr1 ∗ -
– kubectl get all –n je	enkins						
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE		
service/jenkins	ClusterIP	10.100.27.49	<none></none>	8080/TCP	6m20s		
service/jenkins-agent	ClusterIP	10.100.23.241	<none></none>	50000/TCP	6m20s		
NAME	READY	AGE					
<pre>statefulset.apps/jenkins</pre>	s 0/0	6m20s					
🕨 👝 🤹 ≽ 🗢 /VSCode/GitHul	o/ px-jenkin s	s-eks/migrations	👌 🖶 🔑 master	!2 ?5		🖌 < 4s 🗷 <	px-jenkins-dr1 🔹 —

Notice that the Jenkins StatefulSet controller has no instances running. Everything is now replicated to the DR cluster, but it has not been started. Successive replications will keep these objects and the persistent volumes updated based on your replication interval. Also, at any time, you can suspend the replication and scale up the StatefulSet controller to test the viability of your DR site. See the Solution Testing and Validation section for more information.

Install and Configure PX-Backup

Installation

First, create a StorageClass object for the PX-Backup³ database and other components using the following manifest. Copy this into an editor and save it as px-backup-sc.yaml and then applying it to your cluster.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
```

³ Portworx PX-Backup is a separately licensed product. PX-Backup can be used with any CSI storage solution, including AWS's EBS CSI driver. Please consult with Portworx or AWS for more details.

```
name: px-backup-sc
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "2"
  io_profile: auto
allowVolumeExpansion: true
```

To install PX-Backup to your environment, open <u>https://central.portworx.com</u> and log in to your account. In the Portworx spec generator, select **PX-Backup**, and then scroll to the bottom and click **Next**.



Then, on the Spec Details page, under **Select your environment**, choose **Cloud**. In the **Storage Class Name** field, enter "px-backup-sc" from the manifest above that was applied to the cluster. If you are using OpenID Connect (OIDC), you can configure the connection here, as well as if you have a custom registry with the PX-Backup images. Click **Next** to proceed.

PX-Backup		
Spec Details Complete		* required
Namespace *	px-backup	
Install Using	💿 Helm 3 📄 Helm 2	
Select your environment	Cloud OnPrem	
Configuration		
	✓ Use storage class ⑦	
Storage Class Name	px-backup-sc	
	Use your OIDC ⑦	
Custom Registry		
	Use custom registry ⑦	
Reset		Back Next

On the next screen, you are provided with the Helm commands to install PX-Backup. Since you installed Helm before installing Jenkins, you are ready to issue the commands.

- 1. First, copy the contents of "Step 1" and apply it to your production cluster: helm repo add portworx http://charts.portworx.io/ && helm repo update
- 2. Now, copy the contents of the left box in "Step 2" and apply it to your production cluster. This will install PX-Backup in the px-backup namespace. The namespace will be created in the process.

helm install px-backup portworx/px-backup --namespace px-backup --create-namespace --version 1.2.3 -set persistentStorage.enabled=true,persistentStorage.storageClassName="px-backup-sc"

You can monitor the progress of the installation by issuing watch kubectl get pods -n px-backup and wait until all pods have reached a running state. Once everything is up and running, you can port-forward to the UI service and access the GUI. The initial login is admin/admin. Change the password on the first login.

Configure PX-Backup

Once you are logged into the UI for PX-Backup, there are a couple of steps to take to be ready to start protecting your environment.

- First, add your cloud credentials to PX-Backup so it can access both your cluster resources and the S3 bucket used to store the backup objects. To configure your credentials, click Settings in the upper-right corner and choose Cloud Settings.
- On the Cloud Settings page, in the top section, add the IAM Identity's AWS Cloud account credentials used with the Disaster Recovery configuration. Click + Add in the upper-right corner to enter your credentials. Click Add at the bottom once the page is complete.

Add Cloud Account			*required
② Please choose a cloud provider *	AWS / S3 Compliant Objŧ		
② Cloud Account Name *	aws-cloud		
⑦ Access Key *	XXXXXXXXXXXXXX		
🕐 Secret Key *			
🔶 Back		Cancel	Add

Add Backup Location		*required
Name*	portworx-backups	
Cloud Account*	aws-s3-east1 ~	
	S3 Compliant Object Store	
⑦ Path / Bucket *	portworx-backups	
Encryption Key		
Region*	us-east-1	
Endpoint*	s3.amazonaws.com	
	Disable SSL ()	
③ Storage Class	Standard 🗸 🗸 🗸	
← Back		Add

3. Next, click Backup Location on the Cloud Settings page. Click + Add in the lower-right corner.

- 4. Fill out the form. The fields with asterisks (*) are required.
 - a. Name: Provide a name for the backup location.
 - b. Select your cloud account from the drop-down list.
 - c. Provide a bucket name. If the bucket does not exist, it will be created with private access permissions.
 - d. Enter an encryption key if you want to encrypt your backups.

IMPORTANT: Be sure to save the encryption key for restores and backup validation tasks.

- e. Enter the region that your production cluster is in.
- f. The endpoint will be pre-populated.
- g. Leave SSL enabled (do not select Disable SSL).
- h. Select **Standard** for the storage class.
- i. Click Add to finish.

If you want to keep monthly or extended retention backups in Standard-IA, create a second backup location to use with the job definitions.

Once complete	, your cloud	settings	should	resemble	this:
---------------	--------------	----------	--------	----------	-------

Cloud Settings		
Cloud Accounts		+ Add
	aws-s3-east1	
	《 〈 Page 1 of 1	
Backup Locations		+ Add
	aws-s3-pxbackup	
	≪ < Page 1 of 1	

Next, configure your backup schedules.

1. Return to the main screen and click **Settings** again, but select **Schedule Policies**. Then, on the Schedule Policies page, click the plus (+) sign to add a new schedule policy.

Add Policy				×	
Policy Name*					
nightlybackups					
Туре					
Daily				\sim	
Hours*	Minutes*		AM/PM*		
1	0		AM		
Retain ⑦ 7 😓		Incrementa	l Count ③	6 -	
			Cancel	Create	

Provide a policy name and select the type of policy from the Type drop-down list. Above is an example of a nightly backup.
 You can configure a full GFS Backup schedule or a schedule that meets your organization's backup requirements. It might be necessary to create multiple schedules to implement long-term retention for compliance or regulatory reasons.

- 3. Next, configure any pre-backup and post-backup rules that you want to use. Example rules are available at https://backup.docs.portworx.com/use-px-backup/backup-stateful-applications/, including rules for Jenkins. If you have jobs that might be running, add the pre- and post-backup rules for Jenkins based on the reference found here: https://backup.docs.portworx.com/use-px-backup/backup-stateful-applications/, including rules for Jenkins. If you have jobs that might be running, add the pre- and post-backup rules for Jenkins based on the reference found here: https://backup.docs.portworx.com/use-px-backup/backup-stateful-applications/jenkins/.
- 4. Finally, add your production cluster to PX-Backup to begin protecting your Jenkins environment. To do this, click Add Cluster in the upper-right corner. On the resulting screen, fill in the information as described.

Add Kubernete	s Cluster	
K8s Cluster Details		*required
Cluster name		
To get kubeconfig output, use command	"kubectl config viewflattenmini	ify" 🛑
Kubeconfig	* Paste kubeconfig (output from terr	
	— or —	
		Browse
Kubernetes Service	* 🗸 EKS GKE Others	
Cloud Account	* Please select	\sim
	Mana	age Credentials 🛛 🔀
	I Please ensure Stork 2.4+ is running on this clu	uster.
	Copy command to install stork for: 🏾 🌗 PX Cluster	🕼 Non PX Cluster
🗲 Back		Cancel Submit

Once you complete this form with the information for your production cluster and click **Submit**, you will be returned to the PX-Backup main page and your cluster will appear with a green dot next to its name.

• px-aws-prod1	4	17.85 GiB	v1.20.4-eks-6b7464	:
Rows on page: 10 V Showing 1 - 1 of 1			< Page 1 of 1 >	

Configure Your First Backup Job

To configure your first backup job, click the name of the cluster and you will be taken to the job configuration screen.



Here, you can select the namespace(s) to back up from. Once a namespace has been selected (you can select all namespaces if you want), then you can also filter by labels or resource types. However, for Jenkins, it is best to back up everything in the namespace. So, select the Jenkins namespace.

px-	aws-pro	od1 -						
Арр	olications	Backups	Restores	Sche	edules			
jenl	kins	Select Ty	/pe	~	Find labe Q			🕾 Backup
	jenkins-sche age	dule- jenk	ins		component:jenkins-cont + name:jenkins +	Instance:jenkins	Role	í
	jenkins-casc- reload	- jenk	ins		component:jenkins-cont + name:jenkins +	Instance:jenkins	Role	(i)
	jenkins-watc config	:h- jenk	ins		component:jenkins-cont + name:jenkins +	instance:jenkins	RoleBinding	í
	jenkins-sche age	dule- jenk	ins		component:jenkins-cont + name:jenkins +	instance:jenkins	RoleBinding	ì
	jenkins	jenk	ins		component:jenkins-cont + name:jenkins +	Instance:jenkins	StatefulSet	(1)
	kube-root-ca	a.crt jenk	ins				ConfigMap	í
	jenkins-tests	s jenk	ins				ConfigMap	í
					componentienkins-cont +	Instance lenkins		

Scroll down the page and you will see that all the various types of Kubernetes objects in the namespace are listed. The PVCs and PVs are typically near the bottom. Be sure everything is selected, and then click **Backup**.

In the Backup window, you will provide a name, and then you can select the necessary options based on your desired backup schedule. Once you have the desired backup settings, click **Create** to finish setting up the scheduled backups.

TECHNICAL WHITE PAPER

Here is a sample configuration that includes a couple of labels added to the backup for later searches and filters:

CLUSTER:	px-aws-pro	od1	
Enter name for Backup*			
jenkins-nightly			
Backup location 🛈 * 🗹		CSI Snapshot Class ①	
aws-s3-pxbackup		Default	
		You do not have a CSI Si	napshot class.
Now ①	🖉 On a sche	dule 🛈	
Choose a Schedule Policy (]* ⊿		
nightly-backup			\sim
Daily at 01:00AM (retain 7)			
Pre-exec rule 🛛		Post-exec rule 🛛 🗹	
Please select		Please select	
Backup Labels			
× app = jenkins	× schedu	le = nightly	
× type = full-names	pace		
NAMESPACES			
jenkins			
RESOURCES			
 > ClusterRoleBinding (> ConfigMap (5) > PersistentVolume (4) > PersistentVolumeCla > Role (2) > RoleBinding (2) > Secret (2) > Service (2) > Service (2) > ServiceAccount (1) > StatefulSet (1) 	1)) iim (4)		
		Cancel	Create

Once you complete the schedule, it will start. Once complete, it will appear with a green icon on the All Backups page or on the Backups page after clicking into the cluster from the main page.

px-aws-prod1 ~							
Applications Backups Restor	res Schedules						
🛗 May 31, 2021 — Jun 14, 2021							
jenkins-4hr-bu-8c43a		aws-s3- pxbackup	jenkins	1.01 GiB	4	14 Jun 2021, 04:49 PM	
jenkins-4hr-bu-8c43a		aws-s3- pxbackup	jenkins	1.17 GiB	<u>4</u>	14 Jun 2021, 12:49 PM	
jenkins-4hr-bu-8c43a		aws-s3- pxbackup	jenkins	208 MiB	4	14 Jun 2021, 08:49 AM	
jenkins-4hr-bu-8c43a		aws-s3- pxbackup	jenkins	24.63 MiB	4	14 Jun 2021, 04:49 AM	

Here you can see an example of a schedule that runs every four hours, with the most recent backup on top.

In the event you want to test your backups, or restore your environment for whatever reason, simply click the three stacked dots at the end of the line for the backup you want to use, and then select **Restore**.

Enter name for restore*		Choose des	tination cluster 🗊*	
testrestore		px-aws	-prod1	
Default restore	Custom	restore 🛈		
			All resources ir	n all group
> ClusterRoleBinding (1)				🗸 A
> ConfigMap (4)				🗸 A
> PersistentVolumeClaim (4)				🗸 🗸
> Role (2)				🗸 🗸
> RoleBinding (2)				🗸 🗸
> Secret (2)				🗸 🗸
> Service (2)				🗸 🗸
> ServiceAccount (1)				🗸 🗸
> StatefulSet (1)				✓ A

Enter a name and select your production cluster. You can also add your DR cluster and restore to it as well. If you are simply testing your backups, select **Custom restore** and restore to a different namespace. If you need to recover with the backup, use the default restore option, but click the **Replace existing resources** checkbox at the bottom. Once you have the restore configured, click **Restore** to start.

The restore process can be monitored from the Restores tab. Once it finishes, you would find everything in place in the new namespace, if you did a test restore. You can port forward to the Jenkins service and log in to verify that your restore was successful.

Solution Testing and Validation

Functional Testing: Jenkins Controller HA Failover Test

The current state of Jenkins deployment is as follows:

> kubectl NAME jenkins-0	get pods READY 2/2	-n jenkins STATUS Running	-o wide RESTARTS 0	AGE 3d15h	IP 10.28.24.51	NODE ip-10-28-11-8	37.ec2.inter	nal	NOMINATED <none></none>	NODE RE <n< th=""><th>ADINESS</th><th>GATES</th></n<>	ADINESS	GATES
kubectl	get pvc	—n jenkins										
NAME		STATUS	VOLUME				CAPACITY	ACCE	SS MODES	STORAGEC	LASS	AGE
jenkins		Bound	pvc–2f05	d999-b39	1–46cc–a4a2–b	dee199c98e9	100Gi	RWX		px-share	dv4–sc	17d
jenkins-p	lugin-dir	Bound	pvc-9a0d	9983-ffa	b-4d7e-b11b-0	7cffb59c574	25Gi	RWO		px-jenki	ns-sc	17d
jenkins-p	lugins	Bound	pvc–72e6	5c45-f91	6–4f83–9c9a–e	ead8f96f443f	25Gi	RWO		px-jenki	ns-sc	17d
sc-config	-volume	Bound	pvc-e477	7535–251	3–4cdf–98c5–8	Ba1e9a3cb869	25Gi	RWO		px-jenki	ns-sc	17d

Describe the Jenkins volume to see replica placement and status:

kubectl pxc vo	lume inspect pvc-2f05d999-b391-46cc-a4a2-bdee199c98e9
Volume:	662147163463782487
Name:	pvc-2f05d999-b391-46cc-a4a2-bdee199c98e9
Pvc Name:	jenkins
Namespace:	jenkins
Size:	100 GiB
Format:	EXT4
HA:	3
IO Priority:	HIGH
Creation Time:	May 21 18:47:45 UTC 2021
Shared:	v4
Status:	UP
State:	on ip-10-28-9-164.ec2.internal
Device Path:	185717ed-734f-464b-af38-0fa9288350cd
Labels:	<pre>sharedv4_mount_options=vers=4.0</pre>
	namespace=jenkins
	pvc=jenkins
	io_profile=auto
	component=jenkins-controller
	instance=jenkins
	repl=3
	name=jenkins
	sharedv4=true
Fastpath:	
Preference: f	alse
Stats:	
Reads:	36694
Reads MS:	15679
Bytes Read:	247021568
Writes:	723783
Writes MS:	676970
Bytes Written:	7128477696
IOs in progres	is: 0
Bytes used:	979 MiB
Replication Stat	tus: UP
Replica sets on	nodes:
Set: 0	
Node: ip-10	–28–49–141.ec2.internal (Pool 0)
ip-10	–28–81–112.ec2.internal (Pool 0)
ip-10	–28–9–164.ec2.internal (Pool 0)
Pods:	
- Name:	jenkins-0 (113611b2-1295-4cbd-913f-33206a7b53b6)
Namespace:	jenkins
Running on:	ip-10-28-11-87.ec2.internal
Controlled b	y: jenkins (StatefulSet)

To view Jenkins jobs:

						Zadd description
All	+					
s	w	Name ↓	Last Success	Last Failure	Last Duration	Fav
\odot	XÔX	Maven-Test	9 min 42 sec - #440	3 hr 39 min - #426	17 sec	ی (ی
\odot	XÔX	Test Pipeline	5 min 42 sec - #221	N/A	12 sec	
\otimes	ፍ	Twitterbot	4 days 17 hr - #38	1 day 1 hr - #271	1 min 24 sec	ی (ی
Icon: SI	ML			Legend 新 Atom feed for all	ର Atom feed for failures	র Atom feed for just latest builds

Jenkins HA Failover

In the above output, Jenkins pods are running on host: IP-10.28-11-87.ec2.internal

To cordon the cordon node:

```
kubectl cordon ip-10-28-11-87.ec2.internal
node/ip-10-28-11-87.ec2.internal cordoned
```

To delete a Jenkins pod to force failover:

kubectl delete po/jenkins-0 -n jenkins								
pod "jenkins-0" deleted								
kubectl get pods -n jenkins								
NAME READY STATUS R	RESTARTS AGE							
jenkins-0 0/2 Init:0/1 0	0 10s							
kubectl get pods -n jenkins								
NAME READY STATUS REST	TARTS AGE							
jenkins-0 2/2 Running 0	2m53s							
kubectl get pods -n jenkins -o wide								
NAME READY STATUS RESTARTS AG	GE IP	NODE	NOMINATED NODE	READINESS GATES				
jenkins-0 2/2 Running 0 3m	n43s 10.28.75.180	ip-10-28-89-113.ec2.internal	<none></none>	<none></none>				

Note that Jenkins is now running on host: IP-10-28-89-113.ec2.internal, and that failover time is based on pod start-up configuration. Once the pod is available again, the build history is intact and the jobs are still populated and executing on the set schedule as defined.

Build History of Jenkins

	Jun 6	Jun 7	Jun 8	Jun 9		Jun 10	Jun 11
o Ma	aven-Test #429 OTest Pipe	line #217 o Maven-Test #435	Test Pipeline #220	Maven-Test #441			
Test Pip	eline #215 Maven-Test #431	Test Pipeline #218 N	laven-Test #437 OTest	Pipeline #221			
	o Test Pipeline #216	iven-Test #433 OTest Pip	oeline #219 @Maver	n-Test #440			
	Maven-Tes	t #432 @ Maven-Te	est #436 @ Maven-Test #4	139			
	Maven-Test #430	Maven-Test #434	Maven-Test #438				
3							
© SIM							
Timeline	11hr	12hr	13hr	14hr	15hr	16hr	17hr
	Build		Time Since \uparrow		Status		
	Maven-Test #441	I	7 min 10 sec		stable		
	Test Pipeline #22	1	18 min		stable		
0	Maven-Test #440)	22 min		stable		
0	Maven-Test #439)	37 min		stable		

0	Test	t Pipeline #220		48 min	stable			
0	May	Maven-Test #438 5		52 min stable				
0	May	Maven-Test #437 1		1 hr 7 min stable		stable		
0	Test Pipeline #219 1		1 hr 18 min	hr 18 min stable				
0	Maven-Test #436		1 hr 22 min stable					
	May	ven-Test #435		1 hr 37 min	stable			📕 น นธระกฎแบก
All +								
s	w	Name ↓	Last Success	Las	t Failure	Last Duration		Fav
\odot	IÔI	Maven-Test	11 min - #441	3 hr	56 min - #426	18 sec	ø	\overleftrightarrow
\oslash	IÔI	Test Pipeline	22 min - #221	N/A		12 sec	ø	
\otimes	\widehat{m}	Twitterbot	4 days 17 hr - #38	1 da	ıy 1 hr - #271	1 min 24 sec	ø	
Icon: SM	L							

Legend

Atom feed for all

Atom feed for failures

Atom feed for just latest builds

Portworx Backup and Recovery UI

The PX-Backup UI looks like this:

					Settings ~ + Add Cluster
	4 Protected Apps	15.81 GiB Protected Data ①	3 Scheduled Backups		
					> All backups
• px-aws-prod1			15.81 GiB	v1.20.4-eks-6b7464	:
Rows on page: 10 ~ Showing 1 - 1 of 1					< Page 1 of 1 > >>

Here is the Schedule Policies interface.

TECHNICAL WHITE PAPER

Schedule Policies			
bu-4hr-interval	Periodic	Every 4 hour(s) (retain 10)	
hourly-backup	Periodic	Every 1 hour(s) (retain 11)	
nightly-backup	Daily	Daily at 01:00AM (retain 7)	
Backups will be triggered at specified time on application cluster.			
Rows on page: 10 V Showing 1 - 3 of 3			<pre>« < Page 1 of 1 > »</pre>

Cloud accounts and backup location can be found in Cloud Settings:

Cloud Settings				
Cloud Accounts				
		aws-s3-east1		
			<pre>« < Page 1 of 1 ></pre>	
Backup Location				
		aws-s3-pxbackup		
			≪ < Page 1 of 1 >	»

The catalog of previous backups is displayed in All backups.

All	back	ups						< PX-Bac	kup Dashb	board
Fir										
		jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins			08 Jun 2021, 08:48 AM		
		jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins	410 MiB		08 Jun 2021, 04:48 AM		
		jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins	410 MIB		08 Jun 2021, 12:48 AM		
		harbor-nightly-f3aa4	px-aws-prod1	aws-s3-pxbackup	harbor	109 MIB		07 Jun 2021, 09:00 PM		
		jenkins-nightly-bu-d	px-aws-prod1	aws-s3-pxbackup	jenkins			07 Jun 2021, 09:00 PM		
BAC	(A)	jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins	410 MIB		07 Jun 2021, 08:48 PM		
	<u>ه</u>	jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins	549 MIB		07 Jun 2021, 04:47 PM		
		jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins			07 Jun 2021, 12:47 PM		
		jenkins-4hr-bu-8c43a	px-aws-prod1	aws-s3-pxbackup	jenkins			07 Jun 2021, 08:47 AM		
		harbor-nightly-f3aa4	px-aws-prod1	aws-s3-pxbackup	harbor	99.44 MIB		06 Jun 2021, 09:00 PM		
Rows	s on pag	e: 10 ∨ Showing 1 - 10) of 47					« < Page 1	of 5 >	»

Backup and Restore Jenkins

To back up the app, back up the namespace. Note that the additional drop-down and label selector allows for more finegrained backups.

Backup

To back up the application, select the **Backup** button and the dialog box below will appear. Provide a name, select a location, and optionally select a schedule. You will also want to select the pre- and post-backup rules you created if needed. Then click **Create** to start the backup.

Notice that all Kubernetes Objects listed will be part of the backup. The target location is in an S3 bucket.

Create Backup			×
CLUSTER: Enter name for Backup*	px-aws-pi	rod1	
demo-jenkins-bu			
Backup location ①* 🖸		CSI Snapshot Class ①	
aws-s3-pxbackup		Default	
		You do not have a CSI Sna	apshot class.
🔷 Now 🛈	🔵 On a sch	nedule ①	
Pre-exec rule 🛛		Post-exec rule 🛛 🗹	
Please select		Please select	
Backup Labels			
× app = jenkins	× purpo	ose = demo	
NAMESPACES			
jenkins			
RESOURCES			
> ClusterRoleBinding (
ConfigMap (5) Persistent Volume (4)			
PersistentVolume(4) PersistentVolumeCla	im (4)		
> Role (2)			
> RoleBinding (2)			
> Secret (2)			
Service (2) Service (2)			
StatefulSet (1)			
		Cancel	Create

Once the backup is started, it will appear in the backup catalog.

Applicati	ons Backups Ri	estores Schedules					C
🛗 May 25,	2021 — Jun 8, 2021						
	demo-jenkins-bu		aws-s3- pxbackup	jenkins	0 af 0	08 Jun 2021, 10:18 AM	

Clicking the three stacked dots to the right shows the status and configuration of the backup job. You can also select to restore the backup from here.

Once the backup completes, you can view the results and see what was protected.

Note that all Kubernetes objects are listed, including the Persistent Volume and Persistent Volume Claim objects. With this successful backup, you now have multiple options for restoring the application.

Backup Details	×
😗 Restore Backup 🛛 🗎 Delete Backup	
TIMESTAMP 08 Jun 2021, 10:18 AM	
ORIGIN CLUSTER px-aws-prod1	
BACKUP NAME demo-jenkins-bu	
size 1.02 GiB	
BACKUP STATUS - SUCCESS Volumes and resources were backed up successfully	
NAMESPACES — jenkins	
VOLUMES	
 pvc-2f05d999-b391-46cc-a4a2-bdee199c98e9 (980 MiB) pvc-9a0d9983-ffab-4d7e-b11b-07cffb59c574 (104 KiB) pvc-72e65c45-f916-4f83-9c9a-ead8f96f443f (60.50 MiB) pvc-4777535-2513-4cdf-98c5-8a1e9a3cb869 (220 KiB) 	
RESOURCES > ClusterRoleBinding (1) > ConfigMap (4) > PersistentVolume (4) > PersistentVolumeClaim (4)	
 > Role (2) > RoleBinding (2) > Secret (2) > Service (2) > ServiceAccount (1) > StatefulSet (1) 	

If the cluster were to fail, or a mistake is made in applying manifests to the cluster that corrupts or breaks the Jenkins deployment, you can restore it to the current instance. But you can also restore to a different namespace or a different Portworx-enabled Kubernetes cluster.

In the following example, you will restore the Jenkins application to a second cluster located in a different region. The production cluster is in AWS US-East-1. The second cluster is in AWS US-West-1.

The US-West-1 Cluster - Portworx Status and Node List is below:

							_
> kubec	tl pxc pxctl status						
>> Runn	ing pxctl on ip-172-28-0-239.us-west-	-1.compute.int	ernal				
Status:	PX is operational						
License	Trial (expires in 31 days)						
Node ID	890971f9-5a9e-4131-a44a-86e70d866f	:4					
	IP: 172.28.0.239						
	Local Storage Pool: 0 pool						
	POOL IO_PRIORITY RAID_LEVEL	USABLE U	JSED STATI	US ZONE REGION			
	No storage pool						
	Local Storage Devices: 0 device						
	Device Path Media Type Size	e L	.ast–Scan				
	No storage device						
	total – 0B						
	Cache Devices:						
	* No cache devices						
Cluster	Summary						
	Cluster ID: px-jenkins-dr1-61192aae-	-444d-4782-b93	b-dc5f7f634	5f1			
	Cluster UUID: 22189c9a-5cf6-4951-8d1	lc-66debb41e07	/8				
	Scheduler: kubernetes						
	Nodes: 2 node(s) with storage (2 on)	line), 3 node(s) without s	storage (3 online)			
	IP ID		Schee	dulerNodeName	StorageNode	Used	Сар
acity	Status StorageStatus Vers	sion k	kernel	0Ŝ			
	172.28.62.78 c59407e6-889c-4b52-b	6c3-8e7b63ada	a60 ip-1	72-28-62-78.us-west-1.compute.internal	Yes	10 GiB	247
GiB	Online Up	2.7.0.0-0	cee71c 5.4.	117-58.216.amzn2.x86_64 Amazon Linux 2			
	172.28.4.217 c2342107-37a5-4ae9-a	184f-ae8942c31	.565 ip-1	72-28-4-217.us-west-1.compute.internal	Yes	10 GiB	247
GiB	Online Up	2.7.0.0-0	cee71c 5.4.	117-58.216.amzn2.x86_64 Amazon Linux 2			
	172.28.0.239 890971f9-5a9e-4131-a	44a-86e70d866	ofc4 ip-1	72–28–0–239.us-west–1.compute.internal	No	0 B	0 B
Online	No Storage (This node) 2.7.0.0-ccee	71c 5.4.117-5	8.216.amzn2	.x86_64 Amazon Linux 2			
	172.28.36.13 625ad823-68ef-4fc4-b	0486-270f7213f	6a6 ip-1	72–28–36–13.us-west–1.compute.internal	No	0 B	0 B
Online	No Storage 2.7.0.0-ccee	:71c 5.4.117-5	8.216.amzn2	.x86_64 Amazon Linux 2			
	172.28.25.237 2805d1a6-c107-4379-t	o6b9-4bdb0ddf6	a8d ip-1	72–28–25–237.us-west-1.compute.internal	No	0 B	0 B
Online	No Storage 2.7.0.0-cces	e71c 5.4.117-5	8.216.amzn2	.x86_64 Amazon Linux 2			
	10 Storage 2171010-ccee						
Global	Storage Pool						
Global	Storage Pool Total Used : 20 GiB						
Global	Total Used : 20 GiB Total Capacity : 494 GiB						
Global :	Storage Pool Total Used : 20 GiB Total Capacity : 494 GiB tl get nodes						
Global S > kubec NAME	Storage Pool Total Used : 20 GiB Total Capacity : 494 GiB tl get nodes	STATUS RC	ILES AGE	VERSION			
Global () kubec NAME ip-172-	Storage Pool Total Used : 20 GiB Total Capacity : 494 GiB tl get nodes 28-0-239.us-west-1.compute.internal	STATUS RO Ready <r< th=""><th>DLES AGE Ione> 89m</th><th>VERSION v1.19.6-eks-49a6c0</th><th></th><th></th><th></th></r<>	DLES AGE Ione> 89m	VERSION v1.19.6-eks-49a6c0			
Global kubec NAME ip-172- ip-172-	<pre>Storage Pool Total Used : 20 GiB Total Capacity : 494 GiB Il get nodes 28-0-239.us-west-1.compute.internal 28-25-237.us-west-1.compute.internal</pre>	STATUS RO Ready <rr Ready <rr< th=""><th>DLES AGE Ione> 89m Ione> 89m</th><th>VERSION v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0</th><th></th><th></th><th></th></rr<></rr 	DLES AGE Ione> 89m Ione> 89m	VERSION v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0			
Global kubec NAME ip-172- ip-172- ip-172-	<pre>XB-JCONGE 2004 Total Used : 20 GiB Total Capacity : 494 GiB tl get nodes 28-0-239.us-west-1.compute.internal 28-25-237.us-west-1.compute.internal 28-36-13.us-west-1.compute.internal</pre>	STATUS RO Ready <rr Ready <rr Ready <rr< th=""><th>DLES AGE hone> 89m hone> 89m hone> 89m</th><th>VERSION v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0</th><th></th><th></th><th></th></rr<></rr </rr 	DLES AGE hone> 89m hone> 89m hone> 89m	VERSION v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0			
Global : NAME ip-172- ip-172- ip-172- ip-172-	<pre>XB-160-00 20 GB 200-000 XB 200 GB 200 XB 200 X</pre>	STATUS RC Ready <n Ready <r Ready <n Ready <n< th=""><th>DLES AGE hone> 89m hone> 89m hone> 89m hone> 89m</th><th>VERSION v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0</th><th></th><th></th><th></th></n<></n </r </n 	DLES AGE hone> 89m hone> 89m hone> 89m hone> 89m	VERSION v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0 v1.19.6-eks-49a6c0			

The current namespaces in US-West-1 Cluster:

kubectl get nar	nespaces	
NAME	STATUS	AGE
default	Active	100m
kube-node-lease	Active	100m
kube-public	Active	100m
kube-system	Active	100m

To add the US-West-1 Cluster to PX-Backup:

CLUSTER NAME	PROTECTED APPS	PROTECTED DATA
• px-aws-prod1	4	16.82 GiB
• px-jenkins-dr1	0	0 B

PX-Backup can provide backup and recovery services to all connected clusters.

Restore Jenkins Backup

From the UI, click **All Backups >>**. This will present the backup catalog. Here, select the last backup and configure a restore to the remote cluster located in the US-West-1 region.

TECHNICAL WHITE PAPER

In the restore dialog box, provide a name for the restore job. In the destination cluster, both the source and remote clusters are available. The remote cluster is selected and **Custom Restore** is selected to place the restored copy of Jenkins into a different namespace. To proceed, ensure all items are selected and click the **Restore** button.

You can monitor the progress on the resulting page:

Restore Backup "der	no-jenkins-bi	u" (1.02 (_{Choose}	GiB) destination cluster ①*	×
restore-to-west		px-je	enkins-dr1	
Default restore	🥑 Custom	restore 🛈		
Source Namespace(s)*		Des	stination Namespace(s)	
jenkins (1.02 GiB)) – [jenkins-restored	
Search resource nam			 All resources 	s in all groups
ClusterRoleBinding (1) ConfigMan (4)				Al 🗌 Al
 PersistentVolumeClaim 				
Role (2)				🗹 Al
RoleBinding (2)				🗹 Al
Secret (2)				V A
Service (2)				✓ AI
ServiceAccount (1)				
Replace existing resourc	es		Cancel	Restore

The backup was successfully restored to the cluster in the US-West-1 region, as seen below.

<pre>>> kubectl get applic NAMESPACE jenkins-restored</pre>	cationrestore -A NAME restore-to-west-b64e0cc	AGE 24s			
px-jenkins-dr1 ~					
Applications Backups Re	stores Schedules				
📋 May 25, 2021 — Jun 8, 2021					
NAME					
restore-to-west	demo-jenkins-bu	1.11 GiB	<u>4</u>	08 Jun 2021, 05:07 23 PM	
Rows on page: 10 \lor Showing 1 -	1 of 1			« < Page 1 of 1	

The output for kubectl get all -n jenkins-restore looks like this:

<pre>>> kubectl get al NAME pod/jenkins-0</pre>	ll —n je READY 2/2	nkins-resto STATUS Running	ored RESTARTS 0	AGE 4m39s			
NAME service/jenkins service/jenkins-	-agent	TYPE ClusterIP ClusterIP	CLUSTER- 10.100.2 10.100.1	-IP 241.252 132.21	EXTERNAL-IP <none> <none></none></none>	PORT(S) 8080/TCP 50000/TCP	AGE 4m41s 4m41s
NAME statefulset.apps >> kubectl cluste Kubernetes contr CoreDNS is runni dns:dns/proxy	s/jenkin er—info rol plan ing at h	READY s 1/1 e is runnin ttps://7DEF	AGE 4m41s ng at https F44B107F769	5://7DEF4 9CE27BF4B	4B107F769CE27E 3A8DFE8D70.sk1	8F4B3A8DFE8D7 L.us-west-1.e	0.skl.us-west-1.eks.amazonaws.com ks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-

The application is running in the namespace "jenkins-restored" and we can see that the cluster is in US-West-1.

Validate DR Replication Failover Preparedness

Change to the DR Cluster and verify the DR site is usable:

MigrationSchedule jenkins-async-dr suspended successfully ↓ ★ > = ~/VSCode/GitHub/px-jenkins-eks/migrations > ● } master !2 ?5 ↓ kubectx px-jenkins-dr1". ↓ ★ > = ~/VSCode/GitHub/px-jenkins-eks/migrations > ● } master !2 ?5 ↓ ↓ kubectl get all -n jenkins NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE sorvice/jenkins ClusterIP 10 100 27 40 (2000) ↓ \$27 00 (
<pre></pre>	
kubectx px-jenkins-dr1 Switched to context "px-jenkins-dr1". ★ > = ~/VSCode/GitHub/px-jenkins-eks/migrations > = ? master !2 ?5 kubectl get all -n jenkins NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE FORVice/ienkins 10 100 27 40 connes 8000/CEP 6m20c	5
Switched to context "px-jenkins-dr1".	5
<pre>>></pre>	
└─ kubectl get all -n jenkins NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE	
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE	
service/jenking	
Service/ Jenkins Cluster 1 10:100.27:49 <1016> 8060/ 1CF 011205	
service/jenkins-agent ClusterIP 10.100.23.241 <none> 50000/TCP 6m20s</none>	
NAME READY AGE	
statefulset.apps/jenkins 0/0 6m20s	
★ ★ > = ~/VSCode/GitHub/px-jenkins-eks/migrations > = ½ master !2 ?5	h
kubectl scale statefulset/jenkins -n jenkinsreplicas=1	
statefulset.apps/jenkins scaled	
Code/GitHub/px-jenkins-eks/migrations > 6 p master 12 75	h
kubectl get pods -n jenkins	
NAME READY STATUS RESTARTS AGE	
jenkins-0 0/2 Init:0/1 0 11s	
Lubectl get pods -n jenkins	
NAME READY STATUS RESTARTS AGE	
jenkins-0 2/2 Running 0 95s	
A start of the start of	
- Kubecti port-forward svc/jenkins -n jenkins 8080	
Forwarding from 12/.0.0.1:8080 -> 8080	

After suspending the DR migration schedule and scaling up the StatefulSet controller in the DR Cluster, we can then portforward to the Jenkins service and access the application running in the DR Cluster using kubect1 port-forward -n jenkins svc/jenkins 8080.

Next, open your browser and navigate to <u>http://localhost:8080</u>. Your credentials are the same as the production deployment. Once logged in, you can verify that everything is in place and ready in case of a disaster recovery event.

TECHNICAL WHITE PAPER

← → C ☆ ③ localhost:8080	0 e 📄 Portworx	E SE.I	Next 🗎 K8S Info) 🗎 News 🗎 \	/SL » 🗅	🟠 🛛 🛃 🛃 🖉	🕨 🗯 🍘 🗉 Reading	: g List
🧌 Jenkins	Q	search		0	🛕 Je	nkins Admin	→ log o	out
Dashboard →								
管 New Item	All	+				2	add descrip	tion
People	s	w	Name ↓	Last Success	Last Failure	Last Duration		Fav
Build History	\odot	XÔX	Maven- Test	14 min - #562	1 day 10 hr - #426	20 sec	ø	\overleftrightarrow
Service Services	\odot	XÔX	Test Pipeline	40 min - #281	N/A	14 sec	ø	\overleftrightarrow
Lockable Resources	⊗	$\widehat{\psi}$	Twitterbot	6 days 0 hr - #38	2 days 7 hr - #271	1 min 24 sec	ø	\overleftrightarrow
hew View	Icon: S M L			Legend a	Atom feed for all	Atom feed	for failures	
Build Queue	`				<i>ه</i> ۷	tom feed for just la	test builds	
No builds in the queue.								
Build Executor Status	`							
						REST API	Jenkins 2.	293

You can see that the jobs are there and ready to run. When you check the build history, you'll see something like the screenshot below:





Build	Time Since \uparrow	Status	
Maven-Test #563	51 sec	?	
Maven-Test #562	15 min	stable	
Maven-Test #561	30 min	stable	
Test Pipeline #281	41 min	stable	
Maven-Test #560	45 min	stable	
Maven-Test #559	1 hr 0 min	stable	
Test Pipeline #280	1 hr 11 min	stable	
Maven-Test #558	1 hr 15 min	stable	

Not only is the job history there, but the Maven-Test has started on the DR Cluster.

Once done testing the DR site, scale the StatefulSet controller back to 0 and re-enable the migration schedule on the Production cluster:



You can then verify that the DR replication has resumed:



Conclusion

Portworx by Pure Storage is the industry-leading container storage solution. By dynamically providing persistent storage to containers, Portworx enables your company to run your development operations at any scale. Portworx was specifically built from the ground up for containers. Like AWS EKS, which allows you to adapt your compute resources as your needs change, Portworx allows you to "right-size" your storage. No more guesswork or wasted capacity!

Portworx also enables HA across availability zones without the need to deploy multiple Jenkins controllers in an active-passive configuration. Portworx enables you to realize a zero RPO and a recovery time objective (RTO) measured in seconds rather than minutes or longer. By leveraging Portworx for your Jenkins deployments, you can start small and grow as your needs change. Portworx offers unique capabilities, including automated provisioning, dynamic volume expansion, storage pool expansions, data protection, and disaster recovery, and it works alongside EKS to ensure that your applications are always performant, highly available, and protected with the same level of enterprise-class data services as more traditional infrastructures.

Through testing of each component, we have shown that Portworx Enterprise can easily provide the enterprise-grade data services needed to run reliable production systems in the AWS Cloud ecosystem at a large scale. Solving for speed, density, and scale, Portworx not only enables efficient provisioning, cross-AZ high availability, and data that is as mobile as the containers it fuels. Portworx also provides a complete disaster recovery and business continuity solution. Simply add the Disaster Recovery option and enable Metro-DR for zero RPO, or Async-DR for longer distances with a low RPO of 10 minutes. If your business just can't be down, Portworx Enterprise paired with AWS Elastic Kubernetes Service are the tools for the job.

Complete the solution with PX Backup to provide Kubernetes-aware and application-consistent backups. PX-Backup enables enterprises to maintain their data-protection SLAs in this new world of modern applications. Capable of backing up everything from a single container app to distributed systems like Elastic, Cassandra, Kafka, and others, PX-Backup can even back up the entire cluster state to protect your business against the worst of days. Portworx Backup completes a solution delivering a new level of data protection, recoverability, and data mobility for containerized workloads.

Ultimately, AWS EKS and Portworx Enterprise provide a robust and reliable solution for Jenkins CI/CD pipelines. By ensuring that the data is performant, protected, and always available, Portworx Kubernetes Data Services enhances operations in the cloud to a level that no other currently available product can reach. With Portworx Enterprise and AWS EKS, your business can now build software in the cloud without compromise.

Appendix: Additional Testing

Volume Encryption: Cluster-Wide, Test 2.12

```
> kubectl -n kube-system create secret generic px-vol-encryption \
  --from-literal=cluster-wide-secret-key=Il0v3Portw0rX
secret/px-vol-encryption created
> pxctl secrets set-cluster-key --secret cluster-wide-secret-key
>> Running pxctl on ip-10-28-72-236.ec2.internal
Successfully set cluster secret key
> cat << EOF | kubectl apply -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: px-secure-sc
provisioner: kubernetes.io/portworx-volume
parameters:
  secure: "true"
 repl: "3"
EOF
storageclass.storage.k8s.io/px-secure-sc created
> cat << EOF | kubectl apply -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: secure-pvc
spec:
  storageClassName: px-secure-sc
  accessModes:
  - ReadWriteOnce
 resources:
   requests:
      storage: 2Gi
EOF
persistentvolumeclaim/secure-pvc created
> pxctl v l
```

TECHNICAL WHITE PAPER

<pre>> pxctl v l</pre>							
>> Running pxctl on ip-	-10-28-72-236.ec2.internal						
ID	NAME	SIZE	HA	SHARED	ENCRYPTED	PR0XY-V0LUME	<pre>I0_PRIORITYS</pre>
TATUS	SNAP-ENABLED						
941424258947850710	pvc-008381fd-5a4e-41b3-9038-d7a67d989a3b	5 GiB	2	no	no	no	HIGH U
p - attached on 10.28.7	2.236 no						
588388054488095957	pvc-0a445f0e-b9e8-49d4-95fb-669b8b60e158	100 GiB	2	no	no	no	HIGH U
p - attached on 10.28.8	.123 no						
662147163463782487	pvc-2f05d999-b391-46cc-a4a2-bdee199c98e9	100 GiB	3	v4	no	no	HIGH U
p - attached on 10.28.6	0.118 no						
713244019404977633	pvc-3b238398-695b-433a-990f-8399584e4bdf	15 GiB	2	no	no	no	HIGH U
<pre>p - attached on 10.28.6</pre>	7.235 no						
707533177257556091	pvc-45baef3b-dd16-4184-8517-8722c680b3b1	64 GiB	2	no	no	no	HIGH U
<pre>p - attached on 10.28.4</pre>	3.50 no						
650554536075277967	pvc-51e16c22-a543-487c-ad24-7fce7f01e25b	20 GiB	2	v4	no	no	HIGH U
<pre>p - attached on 10.28.6</pre>	7.235 no						
918165325099425006	pvc-6e7cc8cd-7cab-44df-849f-4f416665b01f	2 GiB	3	no	yes	no	HIGH d
own - attached on 10.28	.43.50 no						
515827332276657177	pvc-7100ea08-94bb-40c0-9c48-4c27149ef493	10 GiB	2	no	no	no	HIGH U
<pre>p - attached on 10.28.4</pre>	3.50 no						
258764598894037386	pvc-72e65c45-f916-4f83-9c9a-ead8f96f443f	25 GiB	2	no	no	no	HIGH U
<pre>p - attached on 10.28.6</pre>	0.118 no						
989061513420219150	pvc-86cd7497-3dd6-4808-a479-be4ee23730b5	1 GiB	2	no	no	no	HIGH U
p – attached on 10.28.0	.91 no		_				

Volume Encryption: Volume Granular Encryption, Test 2.13

```
> kubectl -n jenkins-secure create secret generic jenkins-encryption-key --from-literal=secure-
pvc=SuperSecur3Key
secret/ jenkins-encryption-key created
> cat << EOF | kubectl apply -f -</pre>
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: px-secure-sc
provisioner: kubernetes.io/portworx-volume
parameters:
  secure: "true"
  repl: "3"
EOF
storageclass.storage.k8s.io/px-secure-sc created
> cat << EOF | kubectl apply -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: px-secure-rwx-sc
provisioner: kubernetes.io/portworx-volume
parameters:
  sharedv4: "true"
  secure: "true"
  repl: "3"
EOF
storageclass.storage.k8s.io/px-secure-rwx-sc created
> cat << EOF | kubectl apply -f -</pre>
kind: PersistentVolumeClaim
apiVersion: v1
```

```
metadata:
 name: jenkins-data-encrypted
 namespace: jenkins
 annotations:
    px/secret-name: jenkins-encryption-key
   px/secret-namespace: jenkins-encrypted
   px/secret-key: secure-pvc
spec:
  storageClassName: px-secure-rwx-sc
 accessModes:
  - ReadWriteOnce
 resources:
   requests:
      storage: 50Gi
EOF
persistentvolumeclaim/jenkins-data-encrypted created
```

Deploy Jenkins Using Encryption

```
> kubectl apply -f yamls/Jenkins-controller-encrypted.yaml
kubectl get po -n jenkins -w
statefulsets.apps/jenkins-encrypted created
service/jenkins-encrypted created
NAME
                                     READY STATUS
                                                                 RESTARTS
                                                                            AGE
jenkins-0
                      2/2
                              Running
                                                  0
                                                             32m
                     0/2
                              ContainerCreating
jenkins-encrypted-0
                                                  0
                                                             1s
jenkins-encrypted-0 2/2
                              Running
                                                  0
                                                             4s
> kubectl delete secret jenkins-encryption-key -n jenkins
secret "jenkins-encryption-key" deleted
> NODE=`kubectl get pods -l app=jenkins-encrypt -n jenkins -o jsonpath='{.items[0].spec.nodeName}'`
kubectl cordon $NODE
kubectl delete pod jenkins-encrypted-0 -n jenkins
kubectl uncordon $NODE
node/ip-10-28-67-235.ec2.internal cordoned
pod "jenkins-encrypted-0" deleted
node/ip-10-28-67-235.ec2.internal uncordoned
> kubectl describe po jenkins-encrypted-0 -n Jenkins
Name:
             jenkins-encrypted-0
Namespace:
             jenkins
Priority:
             0
```

name=jenkins-encrypted statefulset.kubernetes.io/pod-name=jenkins-encrypted-0 Annotations: kubernetes.io/psp: eks.privileged Status: Error 10.28.75.180 IP: IPs: 10.28.75.180 IP: Controlled By: StatefulSet/jenkins-encrypted Init Containers: init: Container ID: docker://828822c2251e78f69a28c1f1563b8449616fd02d1cc2647c0c8b974945b1ffac Image: jenkins/jenkins:2.293 Image ID: dockerpullable://jenkins/jenkinsasha256:7eafcc2583688b6dd6d8f614bbfe5a67c0d2f8b2ee0563364d5b2f310a74c021 Port: <none> Host Port: <none> Command: sh /var/jenkins_config/apply_config.sh State: Terminated Reason: Error Exit Code: 1 Started: Tue, 08 Jun 2021 09:54:36 -0400 Finished: Tue, 08 Jun 2021 09:54:40 -0400 Ready: True Restart Count: 0 Limits: cpu: 3 memory: 8Gi Requests: 500m cpu: 512Mi memory: Environment: <none> Mounts: /usr/share/jenkins/ref/plugins from plugins (rw) /var/jenkins_config from jenkins-config (rw) /var/jenkins_home from jenkins-home (rw) /var/jenkins_plugins from plugin-dir (rw) /var/run/secrets/kubernetes.io/serviceaccount from jenkins-token-zxhfd (ro) Containers: jenkins: Container ID: docker://35724d04ccea6a8b91aa15acbc3a861d708779ab73aea95dac4224df6ee1c055 jenkins/jenkins:2.293-jdk11 Image: Image ID: dockerpullable://jenkins/jenkinsasha256:439516c825946e9252a0b65049a3bfc500df8d599a57be9bfce08213c837170a Ports: 8082/TCP, 50010/TCP 0/TCP, 0/TCP Host Ports: Arqs:

TECHNICAL WHITE PAPER

```
--httpPort=8080
   State:
                    CrashLoopBackoff
                    Tue, 08 Jun 2021 09:54:51 -0400
     Started:
   Ready:
                    True
   Restart Count: 4
   Limits:
               2
     cpu:
     memory: 4Gi
   Requests:
     cpu:
                500m
               512Mi
     memory:
               http-get http://:8082/login delay=0s timeout=5s period=10s #success=1 #failure=5
   Liveness:
    Readiness: http-get http://:8082/login delay=0s timeout=5s period=10s #success=1 #failure=3
   Startup:
                http-get http://:8082/login delay=0s timeout=5s period=10s #success=1 #failure=12
    Environment:
     POD_NAME:
                                 jenkins-encrypted-0 (v1:metadata.name)
      JAVA_OPTS:
                                 -Dcasc.reload.token=$(POD_NAME)
     JENKINS_OPTS:
                                 50010
     JENKINS_SLAVE_AGENT_PORT:
     JENKINS_UC:
                                 https://updates.jenkins.io
     JENKINS_UC_DOWNLOAD:
                                 https://ftp-nyc.osuosl.org/pub/jenkins
     CASC_JENKINS_CONFIG:
                                 /var/jenkins_home/casc_configs
    Mounts:
      /run/secrets/chart-admin-password from admin-secret (ro,path="jenkins-admin-password")
     /run/secrets/chart-admin-username from admin-secret (ro,path="jenkins-admin-user")
      /usr/share/jenkins/ref/plugins/ from plugin-dir (rw)
      /var/jenkins_config from jenkins-config (ro)
      /var/jenkins_home from jenkins-home (rw)
      /var/jenkins_home/casc_configs from sc-config-volume (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from jenkins-token-zxhfd (ro)
 config-reload:
   Container ID: docker://5e5c6c988e6c325123766d380ffa2ffa53d5db8fd2a6652b34179b0359575520
                    kiwigrid/k8s-sidecar:1.12.0
   Image:
   Image ID:
                    docker-pullable://kiwiqrid/k8s-
sidecarasha256:89739be9ff3894910b29fa505b8726372c8433a6b3786b8e769631d8146b4035
   Port:
                    <none>
   Host Port:
                    <none>
   State:
                    Failed
                    Tue, 08 Jun 2021 09:54:54 -0400
     Started:
   Ready:
                    False
    Restart Count: 0
   Limits:
     cpu:
               500m
     memory: 512Mi
    Requests:
     cpu:
               50m
     memory: 256Mi
   Environment:
```

TECHNICAL WHITE PAPER

```
POD_NAME:
                          jenkins-encrypted-0 (v1:metadata.name)
     LABEL:
                          jenkins-config
                          /var/jenkins_home/casc_configs
     FOLDER:
     NAMESPACE:
                          jenkins
     REQ_URL:
                          http://localhost:8082/reload-configuration-as-code/?casc-reload-
token=$(POD_NAME)
     REQ_METHOD:
                          POST
     REQ_RETRY_CONNECT: 10
   Mounts:
      /var/jenkins_home from jenkins-home (rw)
      /var/jenkins_home/casc_configs from sc-config-volume (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from jenkins-token-zxhfd (ro)
Conditions:
 Type
                    Status
 Initialized
                    False
 Ready
                    False
 ContainersReady
                    False
 PodScheduled
                    True
Volumes:
 plugins:
                PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
   Type:
   ClaimName: jenkins-plugins
   ReadOnly:
               false
 jenkins-config:
               ConfigMap (a volume populated by a ConfigMap)
   Tupe:
   Name:
               jenkins
   Optional: false
 plugin-dir:
   Type:
                PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
   ClaimName: jenkins-plugin-dir
   ReadOnly:
                false
  jenkins-home:
                PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
   Type:
   ClaimName: jenkins
   ReadOnly:
               false
  sc-config-volume:
                PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
   Type:
   ClaimName: sc-config-volume
   ReadOnly:
               false
 admin-secret:
                Secret (a volume populated by a Secret)
   Type:
   SecretName: jenkins
   Optional:
                 false
  jenkins-token-zxhfd:
   Type:
                 Secret (a volume populated by a Secret)
   SecretName: jenkins-token-zxhfd
   Optional:
                false
QoS Class:
                 Burstable
```

Node-Selec	tors: <none></none>			
Toleration	s: node.k	uberne	tes.io/no	t-ready:NoExecute op=Exists for 300s
	node.k	uberne	tes.io/un	reachable:NoExecute op=Exists for 300s
Events:				
Туре	Reason	Age	From	Message
Normal	Scheduled	34m	stork	Successfully assigned jenkins/jenkins-encrypted-0 to ip-10-
28-67-235.	ec2.internal			
Warning	FailedMount	34m	kubelet	MountVolume.SetUp failed for volume "default-token-czw4j" :
failed to	sync secret c	ache:	timed out	waiting for the condition Could not mount volume after
Secret was	deleted			
> kubectl	delete -f jen	kins-e	ncrypted.	Jaml
kubectl de	lete pvc jenk	ins-en	crypted -	n jenkins
statefulse	t.apps "jenki	ns-enc	rypted" d	eleted
persistent	volumeclaim "	jenkin	s-encrypt	ed" deleted

PostgreSQL RWO

Tests: 2.01 through 2.03 Output

```
> kubectl create ns postgres
namespace/postgres created
> kubectl apply -f yamls/postgres-sc.yaml
storageclass.storage.k8s.io/px-postgres-sc created
> kubect1 apply -f yamls/postgres-pvc.yaml
persistentvolumeclaim/postgres-data created
> kubectl get pvc -n postgres
NAME
               STATUS VOLUME
                                                                    CAPACITY
                                                                              ACCESS MODES
STORAGECLASS
                AGE
postgres-data Bound
                      pvc-e27f59af-a275-476c-90f0-3e0d0c681d15
                                                                    2Gi
                                                                               RWO
                                                                                             рх-
postgres-sc 13s
> kubectl label nodes --all node-role.kubernetes.io/worker=true
node/ip-10-28-0-91.ec2.internal labeled
node/ip-10-28-43-50.ec2.internal labeled
node/ip-10-28-60-118.ec2.internal labeled
node/ip-10-28-67-235.ec2.internal labeled
node/ip-10-28-72-236.ec2.internal labeled
node/ip-10-28-8-123.ec2.internal labeled
> # this command assumes your nodes are labled correctly
NODE=`kubectl get nodes -l node-role.kubernetes.io/worker=true -o
jsonpath='{.items[0].metadata.name}'`
> cat << EOF | kubectl apply -f -
apiVersion: portworx.io/v1beta2
kind: VolumePlacementStrategy
```

```
spec:
  replicaAffinity:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - "$NODE"
EOF
volumeplacementstrategy.portworx.io/node-specific created
> cat << EOF | kubectl apply -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: px-node-specific-sc
  labels:
   app: postgres
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "1"
  placement_strategy: "node-specific"
allowVolumeExpansion: true
EOF
storageclass.storage.k8s.io/px-node-specific-sc created
> cat << EOF | kubectl apply -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: vps-test
spec:
  storageClassName: px-node-specific-sc
  accessModes:
  - ReadWriteOnce
  resources:
   requests:
      storage: 20Gi
EOF
persistentvolumeclaim/vps-test created
> # check that the PVC is bound before running the inspect command
kubectl get pvc vps-test
# once it's bound you can inspect it
PVC=`kubectl get pvc vps-test --no-headers | awk '{print $3}'`
pxctl volume inspect $PVC
# and confirm that the IP of the Node in the replica set matches the node's IP
kubectl get node $NODE -o wide
zsh: correct 'pxctl' to 'tpctl' [nyae]? n
NAME
           STATUS
                     VOLUME CAPACITY ACCESS MODES STORAGECLASS
                                                                               AGE
vps-test Pending
                                                         px-node-specific-sc
                                                                               21s
zsh: command not found: pxctl
```

NAME STATUS ROLES VERSION INTERNAL-IP EXTERNAL-IP AGE **OS-IMAGE** KERNEL-VERSION CONTAINER-RUNTIME ip-10-28-0-91.ec2.internal Ready worker 28d v1.19.6-eks-49a6c0 10.28.0.91 3.90.241.207 Amazon Linux 2 5.4.110-54.182.amzn2.x86_64 docker://19.3.13 > kubectl apply -f yamls/postgres-app.yaml deployment.apps/postgres created service/postgres created > kubectl get po -n postgres NAME READY STATUS RESTARTS AGE postgres-ddf7d7dfc-dkgww 1/1 0 15s Runnina > POSTGRES_POD=`kubectl get po -n postgres -l app=postgres -o jsonpath='{.items[0].metadata.name}'` kubectl exec -it -n postgres \$POSTGRES_POD -- psql -c "create database pxdemo;" CREATE DATABASE > kubectl exec -it -n postgres \$POSTGRES_POD -- pgbench -i -s 50 pxdemo NOTICE: table "pgbench_history" does not exist, skipping NOTICE: table "pgbench_tellers" does not exist, skipping NOTICE: table "pgbench_accounts" does not exist, skipping NOTICE: table "pgbench_branches" does not exist, skipping creating tables... 100000 of 5000000 tuples (2%) done (elapsed 0.08 s, remaining 3.84 s) 200000 of 5000000 tuples (4%) done (elapsed 0.17 s, remaining 4.03 s) 300000 of 5000000 tuples (6%) done (elapsed 0.28 s, remaining 4.37 s) 400000 of 5000000 tuples (8%) done (elapsed 0.35 s, remaining 4.06 s) 500000 of 5000000 tuples (10%) done (elapsed 0.43 s, remaining 3.89 s) 600000 of 5000000 tuples (12%) done (elapsed 0.50 s, remaining 3.65 s) 700000 of 5000000 tuples (14%) done (elapsed 0.57 s, remaining 3.50 s) 800000 of 5000000 tuples (16%) done (elapsed 0.64 s, remaining 3.37 s) 900000 of 5000000 tuples (18%) done (elapsed 0.73 s, remaining 3.32 s) 1000000 of 5000000 tuples (20%) done (elapsed 0.82 s, remaining 3.28 s) 1100000 of 5000000 tuples (22%) done (elapsed 0.90 s, remaining 3.18 s) 1200000 of 5000000 tuples (24%) done (elapsed 0.97 s, remaining 3.08 s) 1300000 of 5000000 tuples (26%) done (elapsed 1.06 s, remaining 3.02 s) 1400000 of 5000000 tuples (28%) done (elapsed 1.15 s, remaining 2.96 s) 1500000 of 5000000 tuples (30%) done (elapsed 1.25 s, remaining 2.92 s) 1600000 of 5000000 tuples (32%) done (elapsed 1.36 s, remaining 2.90 s) 1700000 of 5000000 tuples (34%) done (elapsed 1.45 s, remaining 2.82 s) 1800000 of 5000000 tuples (36%) done (elapsed 1.54 s, remaining 2.74 s) 1900000 of 5000000 tuples (38%) done (elapsed 1.63 s, remaining 2.66 s) 2000000 of 5000000 tuples (40%) done (elapsed 1.70 s, remaining 2.55 s) 2100000 of 5000000 tuples (42%) done (elapsed 1.77 s, remaining 2.45 s) 2200000 of 5000000 tuples (44%) done (elapsed 1.85 s, remaining 2.36 s) 2300000 of 5000000 tuples (46%) done (elapsed 1.92 s, remaining 2.26 s) 2400000 of 5000000 tuples (48%) done (elapsed 2.00 s, remaining 2.17 s) 2500000 of 5000000 tuples (50%) done (elapsed 2.08 s, remaining 2.08 s) 2600000 of 5000000 tuples (52%) done (elapsed 2.14 s, remaining 1.98 s) 2700000 of 5000000 tuples (54%) done (elapsed 2.22 s, remaining 1.89 s) 2800000 of 5000000 tuples (56%) done (elapsed 2.29 s, remaining 1.80 s) 2900000 of 5000000 tuples (58%) done (elapsed 2.36 s, remaining 1.71 s)

```
3000000 of 5000000 tuples (60%) done (elapsed 2.43 s, remaining 1.62 s)
3100000 of 5000000 tuples (62%) done (elapsed 2.51 s, remaining 1.54 s)
3200000 of 5000000 tuples (64%) done (elapsed 2.60 s, remaining 1.46 s)
3300000 of 5000000 tuples (66%) done (elapsed 2.69 s, remaining 1.38 s)
3400000 of 5000000 tuples (68%) done (elapsed 2.77 s, remaining 1.31 s)
3500000 of 5000000 tuples (70%) done (elapsed 2.87 s, remaining 1.23 s)
3600000 of 5000000 tuples (72%) done (elapsed 2.96 s, remaining 1.15 s)
3700000 of 5000000 tuples (74%) done (elapsed 3.05 s, remaining 1.07 s)
3800000 of 5000000 tuples (76%) done (elapsed 3.14 s, remaining 0.99 s)
3900000 of 5000000 tuples (78%) done (elapsed 3.22 s, remaining 0.91 s)
4000000 of 5000000 tuples (80%) done (elapsed 3.35 s, remaining 0.84 s)
4100000 of 5000000 tuples (82%) done (elapsed 3.45 s, remaining 0.76 s)
4200000 of 5000000 tuples (84%) done (elapsed 3.54 s, remaining 0.67 s)
4300000 of 5000000 tuples (86%) done (elapsed 3.65 s, remaining 0.59 s)
4400000 of 5000000 tuples (88%) done (elapsed 3.72 s, remaining 0.51 s)
4500000 of 5000000 tuples (90%) done (elapsed 3.80 s, remaining 0.42 s)
4600000 of 5000000 tuples (92%) done (elapsed 3.87 s, remaining 0.34 s)
4700000 of 5000000 tuples (94%) done (elapsed 3.96 s, remaining 0.25 s)
4800000 of 5000000 tuples (96%) done (elapsed 4.05 s, remaining 0.17 s)
4900000 of 5000000 tuples (98%) done (elapsed 4.13 s, remaining 0.08 s)
5000000 of 5000000 tuples (100%) done (elapsed 4.21 s, remaining 0.00 s)
vacuum...
set primary keys...
done.
> kubectl exec -it -n postgres $POSTGRES_POD -- df -m | grep postgres
/dev/pxd/pxd559963210498322444
                                    1952
                                           805
                                                    1030 44% /var/lib/postgresql/data
> kubectl exec -it -n postgres $POSTGRES_POD -- psql pxdemo -c "select count(*) from
pgbench_accounts"
 count
- - - - - - - - -
 5000000
(1 row)
```

Failover Testing: 2.04, Postgres

```
> NODE=`kubectl get pods -l app=postgres -n postgres -o jsonpath='{.items[0].spec.nodeName}'`
kubectl cordon $NODE
kubectl delete pod $POSTGRES_POD -n postgres
node/ip-10-28-8-123.ec2.internal cordoned
pod "postgres-ddf7d7dfc-dkgww" deleted
> kubectl get pod -o wide -n postgres
NAME
                                                                             NODE
                           READY
                                   STATUS
                                             RESTARTS
                                                       AGE
                                                              ΤP
NOMINATED NODE READINESS GATES
postgres-ddf7d7dfc-xgzd8
                          1/1
                                                        23s
                                                              10.28.65.138
                                                                             ip-10-28-67-
                                   Running
                                             0
235.ec2.internal <none>
                                    <none>
> NODE=`kubectl get pods -l app=postgres -n postgres -o jsonpath='{.items[0].spec.nodeName}'`
```

```
kubectl cordon $NODE
kubect1 delete pod $POSTGRES_POD -n postgres
node/ip-10-28-8-123.ec2.internal cordoned
pod "postgres-ddf7d7dfc-dkgww" deleted
> kubectl get pod -o wide -n postgres
NAME
                          READY STATUS
                                                                          NODE
                                         RESTARTS AGE
                                                           TP
NOMINATED NODE READINESS GATES
postgres-ddf7d7dfc-xgzd8 1/1
                                                    23s 10.28.65.138 ip-10-28-67-
                                 Running 0
235.ec2.internal <none>
                                  <none>
> POSTGRES_POD=`kubectl get po -n postgres -l app=postgres -o jsonpath='{.items[0].metadata.name}'`
kubectl exec -it -n postgres $POSTGRES_POD -- df -m | grep postgres
/dev/pxd/pxd559963210498322444
                                 1952 821
                                                1014 45% /var/lib/postgresgl/data
> kubectl exec -it -n postgres $POSTGRES_POD -- psql pxdemo -c "select count(*) from
pgbench_accounts"
 count
----
5000000
(1 row)
> kubectl uncordon $NODE
node/ip-10-28-8-123.ec2.internal uncordoned
```

Read Write Many: Nginix Deployment, 2.05 and 2.06

```
> kubectl apply -f yamls/nginx-sc.yaml
storageclass.storage.k8s.io/px-shared-sc created
> kubectl create ns nginx
kubectl apply -f yamls/nginx-pvc.yaml
namespace/nginx created
persistentvolumeclaim/nginx-pvc created
> kubectl apply -f yamls/nginx-app.yaml
deployment.apps/nginx created
service/nginx-svc created
> kubectl get po -n nginx
NAME
                        READY
                                STATUS
                                          RESTARTS
                                                     AGE
nginx-744b4745dc-w6pgz 1/1
                                Running
                                                     13s
                                          0
> kubectl get svc -n nginx
NAME
           TYPE
                          CLUSTER-IP
                                          EXTERNAL-IP
PORT(S)
              AGE
                                          ac6065bf92048455ba37f6e9cd1b316b-1972197306.us-east-
nginx-svc LoadBalancer 172.20.36.161
1.elb.amazonaws.com 80:30096/TCP 26s
> cat << EOF | cat >> index.html
<html>
<h1>Hello World<\h1>
</html>
```

EUF											
POD=`kubectl get poo	ds -n nginx	no-headeı	rs head -r	n 1 aw	/k '{print \$1}'`						
kubectl cp -n nginx index.html \$POD:/usr/share/nginx/html/index.html											
> kubectl scale depl	<pre>> kubectl scale deploy nginx -n nginxreplicas=3</pre>										
deployment.apps/ngir	nx scaled										
> kubectl get po -n	nginx -owid	е									
NAME	READY	STATUS	RESTARTS	AGE	IP	NODE					
NOMINATED NODE REA	ADINESS GATE	S									
nginx-744b4745dc-8wg	g7w 1/1	Running	0	17s	10.28.37.164	ip-10-28-43-					
50.ec2.internal	<none></none>	<none></none>	>								
nginx-744b4745dc-g2d	c6w 1/1	Running	Θ	17s	10.28.79.33	ip-10-28-72-					
236.ec2.internal	<none></none>	<none></none>									
nginx-744b4745dc-w6p	ogz 1/1	Running	Θ	2m4s	10.28.20.149	ip-10-28-8-					
123.ec2.internal	<none></none>	<none< td=""><td>9></td><td></td><td></td><td></td></none<>	9>								

Volume Snapshot: Test 2.14

PostgreSQL used for testing:

```
> cat << EOF | kubectl apply -f -
apiVersion: volumesnapshot.external-storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: px-postgres-snapshot
 namespace: postgres
spec:
  persistentVolumeClaimName: postgres-data
EOF
volumesnapshot.volumesnapshot.external-storage.k8s.io/px-postgres-snapshot created
> kubectl describe volumesnapshot px-postgres-snapshot -n postgres
Name:
              px-postgres-snapshot
Namespace:
              postgres
Labels:
              SnapshotMetadata-PVName=pvc-e27f59af-a275-476c-90f0-3e0d0c681d15
              SnapshotMetadata-Timestamp=1622737975553645452
Annotations: <none>
API Version: volumesnapshot.external-storage.k8s.io/v1
Kind:
              VolumeSnapshot
Metadata:
  Creation Timestamp: 2021-06-03T16:32:55Z
  Generation:
                       3
  Managed Fields:
    API Version: volumesnapshot.external-storage.k8s.io/v1
    Time:
                  2021-06-03T16:32:55Z
    API Version: volumesnapshot.external-storage.k8s.io/v1
                     kubectl-client-side-apply
    Manager:
```

```
Operation:
                     Update
                     2021-06-03T16:32:55Z
   Time:
 Resource Version: 16829061
 Self Link:
                    /apis/volumesnapshot.external-
storage.k8s.io/v1/namespaces/postgres/volumesnapshots/px-postgres-snapshot
                    c3dbd16b-3bfe-422e-9aa5-482f83287606
 UID:
Spec:
 Persistent Volume Claim Name: postgres-data
 Snapshot Data Name:
                                k8s-volume-snapshot-9790aa14-bf7e-410a-8306-c1915723ad99
Status:
 Conditions:
   Last Transition Time: 2021-06-03T16:32:56Z
   Message:
                          Snapshot created successfully and it is ready
   Reason:
   Status:
                           True
   Type:
                           Ready
 Creation Timestamp:
                           <nil>
Events:
                           <none>
```

Volume Snapshot Restore: Test 2.15

To test the snapshot restore, drop the px-demo database and show you can recover it:

```
> POSTGRES_POD=`kubectl get po -n postgres -l app=postgres -o jsonpath='{.items[0].metadata.name}'`
kubectl exec -it -n postgres $POSTGRES_POD -- psql -c "drop database pxdemo;"
DROP DATABASE
> kubectl exec -it -n postgres $POSTGRES_POD -- psql pxdemo -c "select count(*) from
pgbench_accounts"
psql: FATAL: database "pxdemo" does not exist
command terminated with exit code 2
> cat << EOF | kubectl apply -f -
apiVersion: stork.libopenstorage.org/v1alpha1
kind: VolumeSnapshotRestore
metadata:
 name: postgres-snap-restore
 namespace: postgres
spec:
  sourceName: px-postgres-snapshot
  sourceNamespace: postgres
EOF
kubectl get po -n postgres -w
volumesnapshotrestore.stork.libopenstorage.org/postgres-snap-restore created
```

TECHNICAL WHITE PAPER

NAME	READY	STATUS	RESTARTS	AGE		
postgres-ddf7d7dfc-j28cd	0/1	Pending	Θ	1s		
postgres-ddf7d7dfc-j28cd	0/1	Terminati	ng 0		14s	
postgres-ddf7d7dfc-j28cd	0/1	Terminati	ng 0		14s	
postgres-ddf7d7dfc-qlxd5	0/1	Pending	Θ		0s	
postgres-ddf7d7dfc-qlxd5	0/1	Pending	Θ		0s	
postgres-ddf7d7dfc-qlxd5	0/1	Pending	Θ		10s	
postgres-ddf7d7dfc-qlxd5	0/1	Container	Creating	Θ		10s
postgres-ddf7d7dfc-qlxd5	1/1	Running		Θ		11s
^C%						
> POSTGRES_POD=`kubect1 get	tpo-n	oostgres -	l app=post	gres -	o jso	npath='{.items[0].metadata.name}'`
kubectl exec -it -n postgre	es \$POST	GRES_POD -	- psql pxd	emo -c	c "sel	<pre>ect count(*) from pgbench_accounts"</pre>
count						
5000000						

(1 row)

Volume Resize: Automate with Autopilot, Test 2.15

```
> kubectl get po -n monitor -lapp=prometheus
kubectl get svc -n monitor prometheus
kubectl describe cm -n kube-system autopilot-config
NAME
                                                     READY
                                                             STATUS
                                                                       RESTARTS
                                                                                   AGE
prometheus-prometheus-kube-prometheus-prometheus-0 2/2
                                                                                   28d
                                                             Running
                                                                       1
Error from server (NotFound): services "prometheus" not found
Name:
              autopilot-config
Namespace:
              kube-system
Labels:
              <none>
Annotations: <none>
Data
====
config.yaml:
- - - -
providers:
- name: default
 type: prometheus
  params: url=http://prometheus-kube-prometheus-prometheus.monitor.svc:9090
Events: <none>
> cat << EOF | kubectl apply -f -
apiVersion: autopilot.libopenstorage.org/v1alpha1
kind: AutopilotRule
metadata:
 name: volume-resize
 namespace: kube-system
spec:
  ###### selector filters the objects affected by this rule given labels
  selector:
    matchLabels:
```

```
app: postgres
  ###### conditions are the symptoms to evaluate. All conditions are AND'ed
  conditions:
    # volume usage should be less than 50%
    expressions:
    - key: "100 * (px_volume_usage_bytes / px_volume_capacity_bytes)"
      operator: Gt
      values:
      - "50"
  ##### action to perform when condition is true
  actions:
  - name: openstorage.io.action.volume/resize
   params:
      # resize volume by scalepercentage of current size
      scalepercentage: "100"
      # volume capacity should not exceed 400GiB
      maxsize: "100Gi"
EOF
autopilotrule.autopilot.libopenstorage.org/volume-resize created
> kubectl get events --field-selector involvedObject.kind=AutopilotRule,involvedObject.name=volume-
resize --all-namespaces --sort-by .lastTimestamp
NAMESPACE LAST SEEN TYPE
                                 REASON
                                              OBJECT
                                                                            MESSAGE
default
                       Normal Transition autopilotrule/volume-resize
           1s
                                                                            rule: volume-resize:pvc-
e27f59af-a275-476c-90f0-3e0d0c681d15 transition from Initializing => Normal
```

Scale Up: Adding Storage Using Autopilot

Starting Capacity = 750GB

```
> cat << EOF | kubectl apply -f -</pre>
apiVersion: autopilot.libopenstorage.org/v1alpha1
kind: AutopilotRule
metadata:
 name: pool-expand
spec:
  enforcement: required
  ##### conditions are the symptoms to evaluate. All conditions are AND'ed
  conditions:
    expressions:
    # pool available capacity less than 90%
    - key: "100 * ( px_pool_stats_used_bytes/ px_pool_stats_total_bytes)"
     operator: Gt
     values:
      - "10"
    # pool total capacity should not exceed 2TB
```

```
- key: "px_pool_stats_total_bytes/(1024*1024*1024)"
     operator: Lt
    values:
     - "2000"
 ##### action to perform when condition is true
 actions:
 - name: "openstorage.io.action.storagepool/expand"
  params:
     # resize pool by scalepercentage of current size
    scalepercentage: "50"
     scaletype: "auto"
EOF
> pxctl cluster provision-status
>> Running pxctl on ip-10-28-8-123.ec2.internal
NODE
                                  NODE STATUS
                                               POOL
      POOL STATUS IO_PRIORITY SIZEAVAILABLE USED PROVISIONED
                                                                    ZONE
                                                                                 REGION
             RACK
f7dfb0f8-322c-45fd-aebb-d3cc0322c398 Up 0 ( c4c6dfcb-5069-4c1e-a5f7-8b8ec7c58a70 )
      Online
                               247 GiB 222 GiB
                  HIGH
                                                    25 GiB 665 GiB us-east-1b
      us-east-1
                    default
                                             0 ( dc0a35ed-c6ec-469c-a474-64bc4e38d65f )
185717ed-734f-464b-af38-0fa9288350cd Up
                                247 GiB 226 GiB
      Online
                   HIGH
                                                    21 GiB 648 GiB
                                                                          us-east-1a
      us-east-1 default
c447d117-2ab1-41ec-9b6f-03782334c804 Up 0 ( c986f840-73a4-4b3e-9117-0413dfb8b09f )
                HIGH
                                247 GiB 223 GiB 24 GiB 699 GiB
      Online
                                                                          us-east-1c
      us-east-1
                   default
```

Add Storage Node: Test 3.0.2

```
Status: PX is operational
License: PX-Enterprise extended eval (expires in 59 days)
Node ID: 79bb6bbe-440d-4ac1-9ede-096181bba6ce
IP: 10.28.28.73
Local Storage Pool: 1 pool
POOL IO_PRIORITY RAID_LEVEL USABLE USED STATUS ZONE REGION
0 HIGH raid0 247 GiB 10 GiB Online us-east-1a us-east-1
Local Storage Devices: 1 device
```

Device Path Media Type Size Last-Scan /dev/nvme1n1p2 STORAGE_MEDIUM_NVME 247 GiB 03 Jun 21 18:08 UTC 0:1 247 GiB total Cache Devices: * No cache devices Journal Device: 1 /dev/nvme1n1p1 STORAGE_MEDIUM_NVME Cluster Summary Cluster ID: px-jenkins-prod Cluster UUID: eb9e5c27-dde2-4c08-8577-4b92c97e0e26 Scheduler: kubernetes Nodes: 4 node(s) with storage (4 online), 2 node(s) without storage (2 online) IΡ SchedulerNodeName Auth ΤD StorageNode Used Capacity Status StorageStatus Version Kernel 08 10.28.60.118 f7dfb0f8-322c-45fd-aebb-d3cc0322c398 ip-10-28-60-118.ec2.internal Disabled 25 GiB 247 GiB Online Up 2.7.1.0-Yes 8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.67.235 c447d117-2ab1-41ec-9b6f-03782334c804 ip-10-28-67-235.ec2.internal 24 GiB 247 GiB Disabled Yes Online Up 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 79bb6bbe-440d-4ac1-9ede-096181bba6ce ip-10-28-28-73.ec2.internal 10.28.28.73 Online Up (This node) 2.7.1.0-Disabled 10 GiB 247 GiB Yes 8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.8.123 185717ed-734f-464b-af38-0fa9288350cd ip-10-28-8-123.ec2.internal 21 GiB 247 GiB Online Up Disabled Yes 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 9937ed71-674a-421d-8226-0ae361ba5ebc 10.28.72.236 ip-10-28-72-236.ec2.internal Disabled No 0 B 0 BOnline No Storage 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.43.50 2931ea9b-9ade-4c82-8dfd-5dfcc84d4b0c ip-10-28-43-50.ec2.internal Disabled No 0 B 0 BOnline No Storage 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 Global Storage Pool Total Used : 81 GiB Total Capacity : 988 GiB Upgrade - deploy new version of Portworx - Test 3.03 STC=\$(kubectl get stc -n portworx | awk '{if(NR>1)print \$1}') ٠ kubectl patch stc \$STC -n portworx --type json --patch '[{"op": "replace", "path": "/spec/image", "value": "portworx/oci-monitor:2.7.1"}]' Events: Туре Reason Aae From Messaae ------------- - - -_ _ _ _ _ _ _ _ Normal Pulling 15m kubelet Pulling image "portworx/oci-monitor:2.7.1" Normal Pulled 15m kubelet Successfully pulled image "portworx/ocimonitor:2.7.1" in 4.492196242s Normal Created 15m kubelet Created container portworx Normal Started 15m kubelet Started container Portworx

Storage Rebalance: Automated by Autopilot

This is not shown due to the small amount of data in the cluster.

OS Patch: Upgrade OS in Rolling Fashion, Test 3.06

Current state:

EKS > Clusters > px-aws-prod1													
px-aws-prod1			⊘ Active C	Delete cluster									
① A new Kubernetes version is availa Learn more ²	ble for this cluster.		[Update now									
New AMI release versions are available for 2 Node Groups. Learn more													
Overview Workloads Con	Overview Workloads Configuration												
Nodes (6) Info													
Q Filter nodes by property or value				< 1 >									
Node name \bigtriangledown	Instance type \bigtriangledown	Node Group ∇	Created \bigtriangledown	Status V									
ip-10-28-28-73.ec2.internal	r5.xlarge	controllers	4 minutes ago	⊘ Ready									
ip-10-28-43-50.ec2.internal	m5.xlarge	agents	May 5th 2021 at 9:23 PM	⊘ Ready									
ip-10-28-60-118.ec2.internal	r5.xlarge	controllers	May 5th 2021 at 9:22 PM	⊘ Ready									
ip-10-28-67-235.ec2.internal	r5.xlarge	controllers	May 5th 2021 at 9:22 PM	⊘ Ready									
ip-10-28-72-236.ec2.internal	m5.xlarge	agents	May 5th 2021 at 9:23 PM	⊘ Ready									
ip-10-28-8-123.ec2.internal	m5.xlarge	agents	May 5th 2021 at 9:23 PM	⊘ Ready									

PXCTL Status:

Status:	PX is o	perational						
License:	: PX-Ent	erprise extended	l eval (expires i	in 59 day	Js)			
Node ID:	f7dfb0	f8-322c-45fd-aeb	b-d3cc0322c398					
	IP: 10.2	28.60.118						
	Local St	torage Pool: 1 p	ool					
	POOL	IO_PRIORITY	RAID_LEVEL	USABLE	USED	STATUS	ZONE	REGION
	Θ	HIGH	raid0	247 GiB	25 GiB	Online	us-east-1b	us-east-1
	Local St	torage Devices:	1 device					
	Device	Path	Media Type		Size		Last-Scan	
	0:1	/dev/nvme1n1p2	STORAGE_MEDIUM_	NVME	247 GiB		03 Jun 21 18:11	UTC
	total		-		247 GiB			
	Cache De	evices:						
	* No co	ache devices						
	Kvdb Dev	vice:						

TECHNICAL WHITE PAPER

Device Path Size /dev/nvme2n1 150 GiB * Internal kvdb on this node is using this dedicated kvdb device to store its data. Journal Device: /dev/nvme1n1p1 STORAGE_MEDIUM_NVME 1 Cluster Summary Cluster ID: px-jenkins-prod Cluster UUID: eb9e5c27-dde2-4c08-8577-4b92c97e0e26 Scheduler: kubernetes Nodes: 4 node(s) with storage (4 online), 2 node(s) without storage (2 online) IΡ ID SchedulerNodeName Auth StorageNode Capacity Status StorageStatus Version Kernel Used 0S f7dfb0f8-322c-45fd-aebb-d3cc0322c398 10.28.60.118 ip-10-28-60-118.ec2.internal Disabled Yes 25 GiB 247 GiB Online Up (This node) 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.67.235 c447d117-2ab1-41ec-9b6f-03782334c804 ip-10-28-67-235.ec2.internal 24 GiB 247 GiB 2.7.1.0-Disabled Yes Online Up 8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.28.73 79bb6bbe-440d-4ac1-9ede-096181bba6ce ip-10-28-28-73.ec2.internal Online Up Disabled 10 GiB 247 GiB 2.7.1.0-Yes 8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.8.123 185717ed-734f-464b-af38-0fa9288350cd ip-10-28-8-123.ec2.internal Disabled Yes 22 GiB 247 GiB Online Up 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 10.28.72.236 9937ed71-674a-421d-8226-0ae361ba5ebc ip-10-28-72-236.ec2.internal Disabled No 0 B 0 BOnline No Storage 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 2931ea9b-9ade-4c82-8dfd-5dfcc84d4b0c 10.28.43.50 ip-10-28-43-50.ec2.internal Disabled No 0 B 0 BOnline No Storage 2.7.1.0-8a9e965 5.4.110-54.182.amzn2.x86_64 Amazon Linux 2 Global Storage Pool Total Used : 81 GiB Total Capacity : 988 GiB

Upgrading Kubernetes via Web Console (Including AMI image to match versions):

s > Clusto	ters > px-aws-	prod I									
x-aws-	-prod1						Ø	Active	C	Delete	cluste
Overview	Workloads	Config	uration								
Cluster c	configuration	Info									
Kubernetes	s version Info				Pla	atform version	1 Info				
1.20					ek	:s.1					
Details	Compute	Networkin	g Ad	d-ons	ek	ation Lo	ogging Edit	Update his	story	Tags Node Gi	roup
Details	Compute roups (2) Info roup ame	Networking Desired size	g Ada An ⊽ ve	d-ons	ek Authentic	s.1	ogging Edit mplate	Update his	story	Tags Node Gi	roup
Details Node Gro Gro nai age	Compute roups (2) Info roup ame	Networking Desired size	g Ada ▼ AM ve 1.2	1-ons 11 release rsion 20.4-20210	ek Authentic ⊽	ation Lo Launch ter eksctl-px-a	ogging Edit mplate ws-prod1-n	Update his Delete	story Add	Tags Node Gi	roup tatus

Portworx Cluster status after update:

kube	ctl pxc	pxctl sta	atus											
>> Runn:	ing pxct	l on ip-:	10-28-81	-112.ec2.interna	al									
Status:		perationa	al											
License	PX-Ent	erprise e	extended	eval (expires :	in 59 day									
Node ID:	c92a66	07-25f9-4	43a7-b38	a-42e06dccfea8										
	IP: 10.	28.81.112	2											
	Local S	torage Po	ool: 1 p	ool										
	POOL	IO_PRIO	RITY	RAID_LEVEL	USABLE	USED	STATUS	ZONE	REGION					
				raid0	247 GiB	10 GiB	Online	us-east-1c	us-east					
	Local S	torage De	evices:	1 device										
	Device	Path		Media Type		Size		Last-Scan						
	0:1	/dev/nvr	meln1p2	STORAGE_MEDIUM	_NVME	247 GiB		03 Jun 21 21:44	\$ UTC					
	total					247 GiB								
	Cache D	evices:												
	* No c	ache dev:	ices											
	KVdb De	vice:	61											
	Device	Path	51Ze											
	/dev/nv	mezni mezlikudi	100 010	a nada ia usina	this dod	iested k	udb daus	en to store its	data					
	lournal	Davica:	o on thr	s node is using	CHIS GEO	ICated K	AUD GEAT	ce to store its	uata.					
	Journa	/dou/our	tolo1o1	CTODACE MEDTIM	NUME									
Cluster	Summarry		lietutht	210KM0E_HED10H	INVINE									
cluster	Cluster		ienkins-	nrod										
	Cluster		9e5c27-	dde2-4c08-8577-4	169269768	e26								
	Schedul	er: kube	rnetes	JUCE 4000 0577	103203700	CLU								
	Nodes:	6 node(s)) with s	torage (6 online	•)									
	IP		ID				Schedul	erNodeName		Auth	Storage	Node	Used	Capa
city	Status	Storages	Status	Version	Kernel			05						
	10.28.4	9.141	f7dfb0fi	8-322c-45fd-aebl	-d3cc032	2c398	ip-10-2	8-49-141.ec2.int		Disabled			25 GiB	247
GiB				2.7.1.	0-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2				
	10.28.8		c92a660	7-25f9-43a7-b38	-42e06dc	cfea8	ip-10-2	8-81-112.ec2.int		Disabled			10 GiB	
GiB		Online		s node) 2.7.1.	0-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2				
	10.28.8	2.219	c447d11	7-2ab1-41ec-9b6	F-0378233	4c804	ip-10-2	8-82-219.ec2.int		Disabled			22 GiB	247
GiB		Online		2.7.1.	9-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2				
	10.28.1	1.87	79bb6bb	e-440d-4ac1-9ed	e-096181b	ba6ce	ip-10-2	8-11-87.ec2.inte	ernal	Disabled	Yes		10 GiB	247
GiB		Online		2.7.1.	8-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2				
	10.28.3	6.85	5dfa752	e-f366-4fac-b4f	e-39554e5	1244e	ip-10-2	8-36-85.ec2.inte	ernal	Disabled	Yes		10 GiB	247
GiB		Online	Up	2.7.1.	0-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2				
0.10	10.28.2	3.56	185717e	d-7341-464b-af3	3-01a9288	350cd	1p-10-2	8-23-56.ec2.inte	ernal	Disabled			22 GiB	
GIB		Unline		2.7.1.0	0-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2				
Global	torage	POOL		10										
	Total U	sed	99 G	10										

Resiliency Testing:

Node Reboot:

This was difficult to capture as the node went offline and came back online without interrupting operations outside of Userspace (non-daemonset); pods were rescheduled within seconds on a different node.

Node	Shute	down:											
Status:	PX is op	perational	l										
License	PX-Ente	erprise ex	ktended eval	l (expires i	n 58 day	s)							
Node ID:	: 5dfa752	2e-f366-41	fac-b4fe-395	554e51244e									
	IP: 10.2	28.36.85											
	Local St	torage Poo	ol: 1 pool										
	POOL	IO_PRIORI	ITY RAII	D_LEVEL	USABLE	USED	STATUS	ZONE	REGION				
		HIGH	raid	10	247 GiB	12 GiB	Online	us-east-1b	us-eas				
	Local St	torage Dev	vices: 1 dev	vice									
	Device	Path	Medi	іа Туре		Size		Last-Scan					
	0:1	/dev/nvme	e1n1p2 STOF	RAGE_MEDIUM_	NVME	247 GiB			1:48 UTC				
	total					247 GiB							
	Cache De	evices:											
	*No ca	ache devi	ces										
	Journal	Device:											
		/dev/nvme	eln1p1 STOF	RAGE_MEDIUM_	NVME								
Cluster	Summary												
	Cluster	ID: px-je	enkins-prod										
	Cluster	UUID: eb9	e5c27-dde2-	-4c08–8577–4	b92c97e0	e26							
	Schedule	er: kubern	netes										
	Nodes: 6	5 node(s)	with storag	ge (5 online)								
	IP		ID .				Schedul	erNodeName		Auth	StorageNode	Used	(
apacity	Status	StorageSt	tatus Vers	sion	Kernel			05					
	10.28.49	9.141	f7dfb0f8–322	2c-45fd-aebb	-d3cc032	2c398	ip-10-2	8-49-141.ec2	.internal	Disabled	Yes	26 GiB	
47 GiB		Online l	Jp	2.7.1.0	-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2			
	10.28.81	L.112 (c92a6607-251	f9–43a7–b38a	-42e06dc	cfea8	ip-10-2	8-81-112.ec2	.internal	Disabled	Yes	11 GiB	
47 GiB		Online l	Jp	2.7.1.0	-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2			
	10.28.82	2.219	c447d117–2at	o1-41ec-9b6f	-0378233	4c804	ip-10-2	8-82-219.ec2	.internal	Disabled	Yes	24 GiB	
47 GiB		Online l	Jp	2.7.1.0	-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2			
	10.28.11	1.87	79bb6bbe-44	0d-4ac1-9ede	-096181b	ba6ce	ip-10-2	8-11-87.ec2.	internal	Disabled	Yes	25 GiB	
47 GiB		Online l	Jp	2.7.1.0	-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2			
	10.28.36	5.85	5dfa752e-f36	66–4fac–b4fe	-39554e5	1244e	ip-10-2	8-36-85.ec2.	internal	Disabled	Yes	12 GiB	
47 GiB		Online l	Jp (This noo	de) 2.7.1.0	-8a9e965	5.4.117	-58.216.	amzn2.x86_64	Amazon	Linux 2			
	10.28.23	3.56	185717ed-734	1f-464b-af38	-0fa9288	350cd	ip-10-2	8-23-56.ec2.		Disabled			ablel
navailab	ole	Offline [Down	2.7.1.0	-8a9e965	5.4.117	-58.216.	amzn2.x86_64					
Global S	Storage F	Pool											

AWS Auto-scaling replaced the node based on my definitions and the node was resynced and back online:



Disk Failure: (simulation)

> NODE=`kubectl get pods -l app=postgres -n postgres -o jsonpath='{.items[0].spec.nodeName}'`
echo \$NODE

ip-10-28-82-219.ec2.internal

> kubectl get pods -n postgres										
NAME	READY	STATUS	RESTARTS	AGE						
nginx-6598b75bbf-zmsfs	1/1	Running	0	123m						
postgres-ddf7d7dfc-slrsq	1/1	Running	0	123m						

Node IP-10-28-82-219 placed into maintenance mode to simulate a disk failure:

Cluster	1 /dev/nvr Summary Cluster ID: px- Cluster UUID: el Scheduler: kube Nodes: 6 node(s	meinipi ST jenkins-pro p9e5c27-dde rnetes) with stor	TORAGE_MEDI od 22-4c08-857 rage (5 onl	UM_NVME 7-4b92c97e0 ine)	le26					
	IP	ID				SchedulerNode	eName	Auth	StorageNode	Used
apacity	Status	St	orageStatu	s Version		Kernel		0S		
	10.28.49.141	f7dfb0f8-3	822c-45fd-a	ebb-d3cc032	2c398	ip-10-28-49-3	141.ec2.internal	Disabled		25 GiB
47 GiB	Online		Up			0-8a9e965 5.4.2	117-58.216.amzn2.x8	36_64 Am	azon Linux 2	
	10.28.81.112	c92a6607-2	25f9-43a7-b	38a-42e06dc	cfea8	ip-10-28-81-3	112.ec2.internal	Disabled	Yes	11 GiB
47 GiB	Online		Up			0-8a9e965 5.4.	117-58.216.amzn2.x8	36_64 Am	azon Linux 2	
	10.28.82.219	c447d117-2	ab1-41ec-9	b6f-0378233	4c804	ip-10-28-82-2	219.ec2.internal	Disabled	Yes	Unavailabl
navaila	ble Online	(StorageDow	/n) In M	aintenance	2.7.1.	0-8a9e965 5.4.2	117-58.216.amzn2.x8	36_64 Am	azon Linux 2	
	10.28.11.87	79bb6bbe-4	40d-4ac1-9	ede-096181b	ba6ce	ip-10-28-11-	87.ec2.internal	Disabled	Yes	25 GiB
47 GiB	Online		Up		2.7.1.	0-8a9e965 5.4.2	117-58.216.amzn2.x8	36_64 Am	azon Linux 2	
	10.28.36.85	5dfa752e-1	366-4fac-b	4fe-39554e5	1244e	ip-10-28-36-	85.ec2.internal	Disabled	Yes	12 GiB
47 GiB	Online		Up (This node)		0-8a9e965 5.4.	117-58.216.amzn2.x8	36_64 Am	azon Linux 2	
	10.28.9.164	185717ed-7	'34f–464b–a	f38-0fa9288	350cd	ip-10-28-9-10	64.ec2.internal	Disabled	Yes	23 GiB
47 GiB	Online					0-8a9e965 5.4.2	117-58.216.amzn2.x8	36_64 Am	azon Linux 2	
Global	Storage Pool									
	Total Used	: 98 GiB								
	Total Capacity	: 1.2 Ti	3							
> kubec	tl get pod –n pos	stgres –o v	/ide							
NAME		RÉADY	STATUS	RESTARTS	AGE	IP	NODE		NOMINATED NODE	READINESS GA
nainx-6	598b75bbf-zmsfs	1/1	Running	0	131m	10.28.37.247	ip-10-28-49-141.e	ec2.internal	<none></none>	<none></none>
postgre	s-ddf7d7dfc-slrs	1/1	Running	0	131m	10.28.67.18	ip-10-28-82-219.0	ec2.internal	<none></none>	<none></none>

Note the PostgreSQL pod was not restarted and is still running on the same node, but using the storage replica on another node. Note also that the database is still available.



Bringing Portworx on the node back online resynchronizes any changes and marks node available:

/ Kubec	cc pxc p	ACT Sta	10 20 11	07									
>> Runn:	ing pxct	cion ip-	10-28-11	-8/.ec2.internal									
Status:	PX 15 C	operation	aι 										
License	PX-Ent	terprise	extended	eval (expires i	n 58 day	S)							
Node ID:	: /9bb6t	be-440d-	4ac1-9ed	e-096181bba6ce									
	IP: 10.	28.11.8/											
	Local	storage P	ool: 1 p	00 L									
	POOL	10_PR10	RITY	RAID_LEVEL	USABLE	USED	STATUS	ZONE	REGION				
	0	HIGH		raid0	247 G1B	25 G1B	Unline	us-east-1a	us-east				
	Local	storage D	evices:	1 device									
	Device	Path		Media Type		Size		Last-Scan					
	0:1	/dev/nv	me1n1p2	STORAGE_MEDIUM_	NVME	247 G1B		03 Jun 21 22:19					
	total					247 G1B							
	Cache L	Devices:											
	* No (cache dev	ices										
	Kvdb De	evice:											
	Device	Path	Size										
	/dev/nv	/me3n1	150 GiB										
	* Inte	ernal kvd	b on thi	s node is using	this ded	icated k	vdb devi	ce to store its	data.				
	Journa	Device:											
61	1	/dev/nv	meinipi	STORAGE_MEDIUM_	NVME								
Cluster	Summary	-											
	Cluster	· ID: px-	jenkins-	prod	-02-07-0	- 26							
	Cluster	OUID: e	D965C2/-	aae2-4c08-8577-4	b92c97e0	e26							
	Nedeau	6 nodo/o	hetes	terrare (6 emline									
	Nodes:	o noue(s) WICH S	torage (6 ontine)		Cabadul			A+h	Ctonnellada	llaad	C
e i tu	1P Status	C+orogo	Status	Versien	Kernel		Schedul	ernouename		Auth	StorageNode	usea	Capa
CITY	Status	Storage	Status	Version	da colora	2.000	10 20	US 8 40 141 ac2 int	o mo 1	Dischlod	Vee		247
C-FP	10.20.4	19.141 Opling		0-322C-4310-debb	-0300052	ZC390 E / 117		0-49-141.002.100	Amazon	DISADIEU		25 GID	247
OID	10 20 0	01 112	op	2./.1.0 7 35f0 43a7 b30a	-0096900	5.4.11/	-30.210.	amznz.xoo_04	Anazon	Dischlod	Vec	11 C-P	247
CiP		Online	Up	2219-43d/-030d	-4200000	E / 117	_50 216	0-01-112.0C2.100	Amazon	Disableu		11 016	247
GID	10 20 0	22 210	op 6447d11	2.7.1.0 7_2ab1_41ac_0b6f	-0036303	J.4.11/	-J0.210.	amznz:x00_04	Anazon	Disabled	Vec	24 CIP	247
CiR		Online	Un Un	7 - Zabi-4iec-9001	-0370233	5 / 117.	_50 216	2727219,002,1110	Amazon	Linux 2		24 GIB	247
OID	10 20 1	11 07	7055655	0.440d_45c1_0odo	_006101b	ba6co	in_10_2	0_11_07_0c2_into	rnal	Dicabled	Voc	DE CIP	247
CHP	10.20.1	Online	/900000	c = 440 u = 4a CI = 3Eu c	-0501010		-E0 216	0-11-07.ecz.inte	Amazon	Disableu		25 616	247
OID	10 20 2		5dfa752	5 1100C/ 2:/:1:0	-2055/05	12446	in_10_2	dm2112.00_04 0-36-05 oc2 into	rnal	Dicabled	Voc	12 CiP	247
GiB	10.20.3	Online	lln	2 7 1 0	-8-0-065	5 / 117.	-58 216	amzn2 v86 64	Amazon	Linux 2		12 010	
010	10.28.0	0.164	1857176	d_734f_464b_af39	_0fa0200	350cd	in-10-2	R_9_164_ec2_inte	rnal	Disabled	Yes	23 G4B	
GiB	1012013	Online	lln	2 7 1 0	-8200065	5 4 117	-58 216	amzn2 x86 64	Amazon	Linux 2		23 01D	247
Global	Storage	Pool		2.7.1.0	-0496903	314111/	3012101	um21121200_04	20102011	CINGX 2			
otobat :	sconage	1000											

Note that during this operation the database was able to respond to queries and was not rescheduled to another node:

> kubectl get pods -n pos	tgres –l	app=postg	res —o wide					
NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
postgres-ddf7d7dfc-slrsq	1/1	Running	0	138m	10.28.67.18	ip-10-28-82-219.ec2.internal	<none></none>	<none></none>

©2021 Pure Storage, the Pure P Logo, and the marks on the Pure Trademark List at https://www.purestorage.com/legal/productenduserinfo.html are trademarks of Pure Storage, Inc. Other names are trademarks of their respective owners. Use of Pure Storage Products and Programs are covered by End User Agreements, IP, and other terms, available at: https://www.purestorage.com/legal/productenduserinfo.html are trademarks of their respective owners. Use of Pure Storage Products and Programs are covered by End User Agreements, IP, and other terms, available at: https://www.purestorage.com/legal/productenduserinfo.html and <a h

The Pure Storage products and programs described in this documentation are distributed under a license agreement restricting the use, copying, distribution, and decompilation/reverse engineering of the products. No part of this documentation may be reproduced in any form by any means without prior written authorization from Pure Storage, Inc. and its licensors, if any. Pure Storage may make improvements and/or changes in the Pure Storage products and/or the programs described in this documentation at any time without notice.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

Pure Storage, Inc. 650 Castro Street, #400 Mountain View, CA 94041

purestorage.com

800.379.PURE





PS2133-01 09/2021 i