Use Case Validations

NetApp Solutions

NetApp
February 15, 2023

Table of Contents

Use-case validation: DevOps with NetApp Astra

The following use cases have been validated for DevOps with NetApp Astra:

- Integrate Protection into CI/CD Pipelines with NetApp Astra Control
- Leverage Astra Control to facilitate Post-mortem Analysis and Restore the Application
- Accelerating Software Development with NetApp FlexClones

Next: Videos and Demos - DevOps with NetApp Astra.

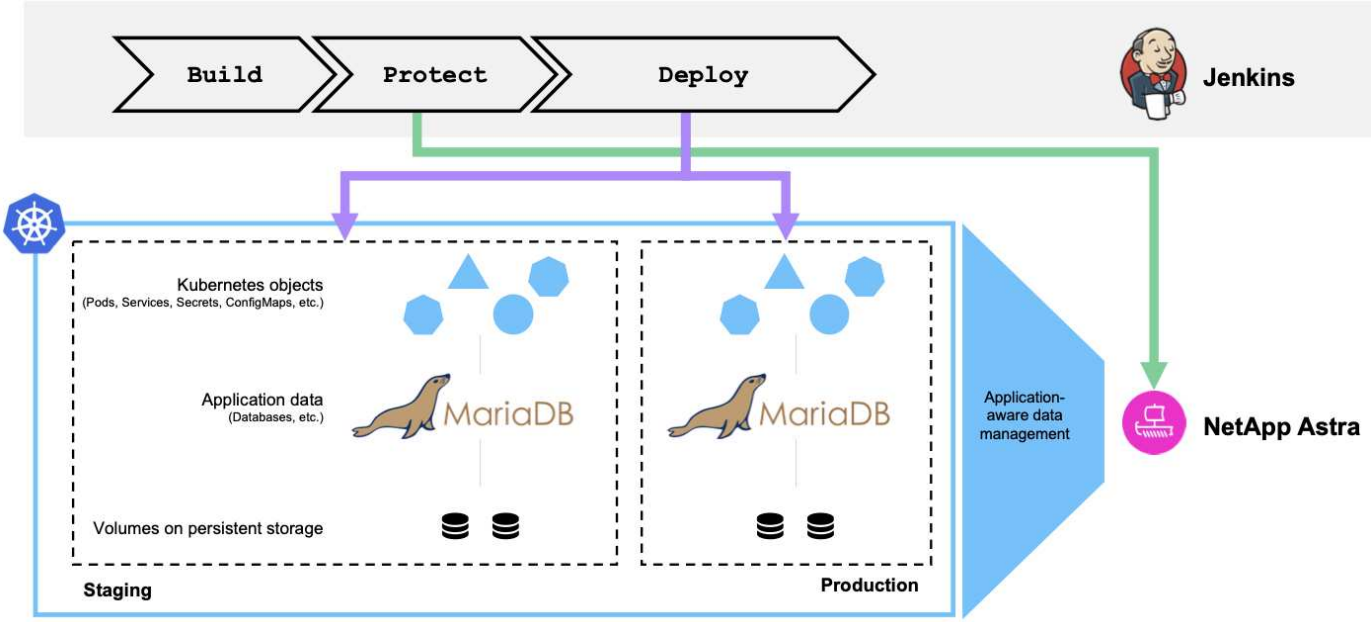
Integrate Protection into CI/CD Pipelines with NetApp Astra Control

Overview

One of the most common uses of DevOps workflows is continuous integration and continuous deployment (CI/CD) pipelines that build, integrate, and run automated test suites on applications as developers commit new code. DevOps engineers and site-reliability engineers (SREs) typically have pipelines dedicated to the various workflows for new feature development, regression testing, bug fixes, quality engineering, and other functions in the development process.

As teams increase their level of automation, the pace of change for in-production applications can feel overwhelming. Therefore, some teams prefer to protect in-production applications or services. In addition to protecting the code and container images, they also want to protect the application state, configuration data (such as Kubernetes objects and resources associated with the application), and an application's persistent data.

In this use case, we take a closer look at a promotion-to-production pipeline that deploys a new version of an application: first into a staging environment and then into a production environment. This example applies equally to the major public clouds and also to an on-premises environment. Although we show the deployment of one version of the app, the pipeline can also be used with other strategies, such as blue/green or canary deployment. As part of the CI/CD pipeline, we're going to protect the application by creating a complete application backup. An application-aware backup of the in-production application and its data, state, and configuration can be useful for numerous DevOps workflows.



The application used for validating this use-case was Magento, an e-commerce solution with a web-based front end; an Elasticsearch instance for search and analysis features; and a MariaDB database that tracks all the shopping inventory and transaction details. This containerized application was installed in a Red Hat OpenShift cluster. Every pod in the application used persistent volumes to store data. The persistent volumes were automatically created by NetApp Astra Trident, the Container Storage Interface–compliant storage orchestrator for Kubernetes that enables storage to be provisioned on NetApp storage systems. Further, to utilize the Astra Control Center's application protection capabilities, the application in question was managed by Astra Control, which was then used to trigger application backups that stored the state of the application along with the data held in persistent volumes. We used the NetApp Astra Control Python SDK to automate the process of triggering application backups, which was then introduced into a CI/CD pipeline. This pipeline was created and executed using a popular CI/CD tool called [Jenkins] to automate the flow to build, protect, and deploy the application.

Let us run through the prerequisites and procedure to introduce protection in a CI/CD pipeline.

Use-case validation prerequisites

The following tools or platforms were deployed and configured as prerequisites:

1. Red Hat OpenShift Container Platform
2. NetApp Astra Trident installed on OpenShift with a backend to NetApp ONTAP system configured
3. A default storageclass configured, pointing to a NetApp ONTAP backend
4. NetApp Astra Control Center installed on an OpenShift cluster
5. OpenShift cluster added as a managed cluster to Astra Control Center
6. Jenkins installed on an OpenShift cluster and configured with an agent node with a Docker engine installed on it

Installing the application

Let's start with the initial installation of the application in the staging and production environments. For the purpose of this use case, this step is a prerequisite, so it is performed manually. The CI/CD pipeline is used for subsequent build and deploy workflows as a result of new version releases of the application.

The production environment in this use case is a namespace called `magento-prod`, and the corresponding staging environment is a namespace called `magento-staging` configured on the Red Hat OpenShift cluster. To intall the application, complete the following steps:

1. Install the Magento application using bitnami helm chart on the production environment. We use RWX PVs for Magento and Mariadb pods.

```
[netapp-user@rhel7 na_astra_control_suite]$ helm install --version 14
magento bitnami/magento -n magento-prod --create-namespace --set
image.tag=2.4.1-debian-10-
r11,magentoHost=10.63.172.243,persistence.magento.accessMode=ReadWriteMa
ny,persistence.apache.accessMode=ReadWriteMany,mariadb.master.persistenc
e.accessModes[0]=ReadWriteMany
```

> Magento bitnami helm chart requires a LoadBalancer service to expose the Magento GUI service. We used MetalLB for providing an on-prem load balancer service in this example.

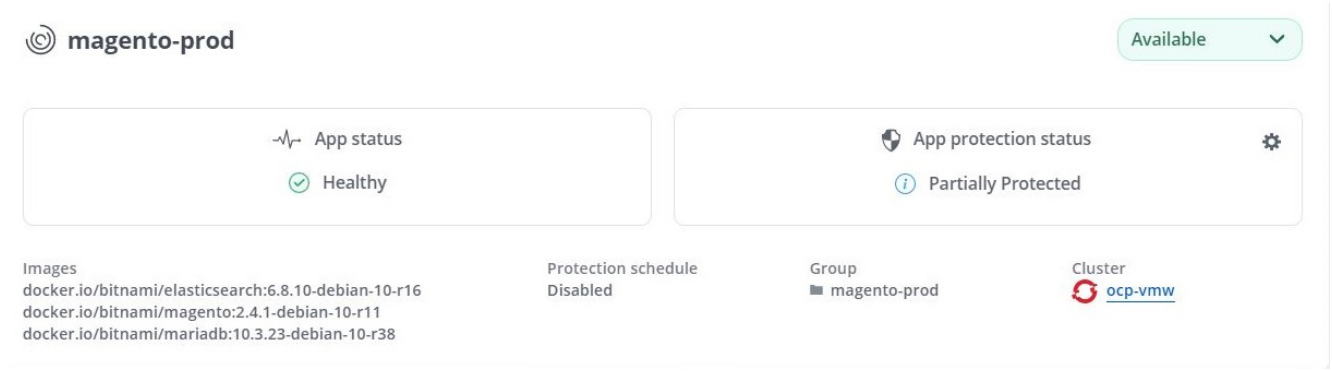
2. After a few minutes, verify that all pods and services are running.

```
[netapp-user@rhel7 na_astra_control_suite]$ oc get pods -n magento-prod
NAME                                                    READY   STATUS
RESTARTS   AGE
magento-9d658fd96-qrxmt                                 1/1     Running
0          49m
magento-elasticsearch-coordinating-only-69869cc5-768rm  1/1     Running
0          49m
magento-elasticsearch-data-0                            1/1     Running
0          49m
magento-elasticsearch-master-0                          1/1     Running
0          49m
magento-mariadb-0                                       1/1     Running
0          49m
```

3. Repeat the same procedure for the staging environment.

Manage the Magento application in Astra Control Center

1. Navigate to Applications and select the Discovered applications tab.
2. Click the ellipsis against the Magento application in the production environment (`magento-prod`), and click Manage.
3. The Magento application is now managed by the Astra Control Center. All operations supported by Astra Control can be performed on the application. Note the version of the application as well.

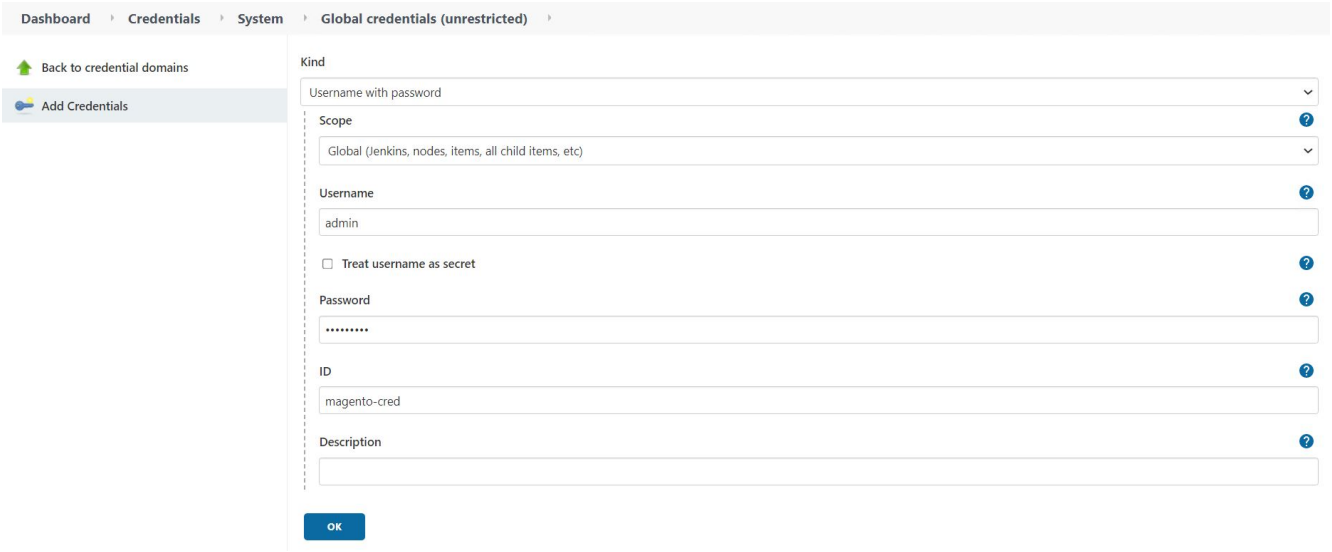


4. Repeat the steps for managing the Magento application in the staging environment (`magento-staging`).

CI/CD pipeline with integrated protection

When we work with new versions of applications, we use a CI/CD pipeline to build the container image, take backups of both the staging and production environments, deploy the new version of the application to the staging environment, wait for approval to promotion to production, and then deploy the new version of the application to the production environment. To use a CI/CD pipeline, complete the following steps:

1. Log into Jenkins, and create the required credentials: one for Magento creds, one for Mariadb admin creds, and the third for Mariadb root creds.
2. Navigate to Manage Jenkins > Manage Credentials and click the appropriate domain.
3. Click Add Credentials, and set the kind to Username with password and scope set to Global. Enter the username, password, and an ID for the credentials and click OK.

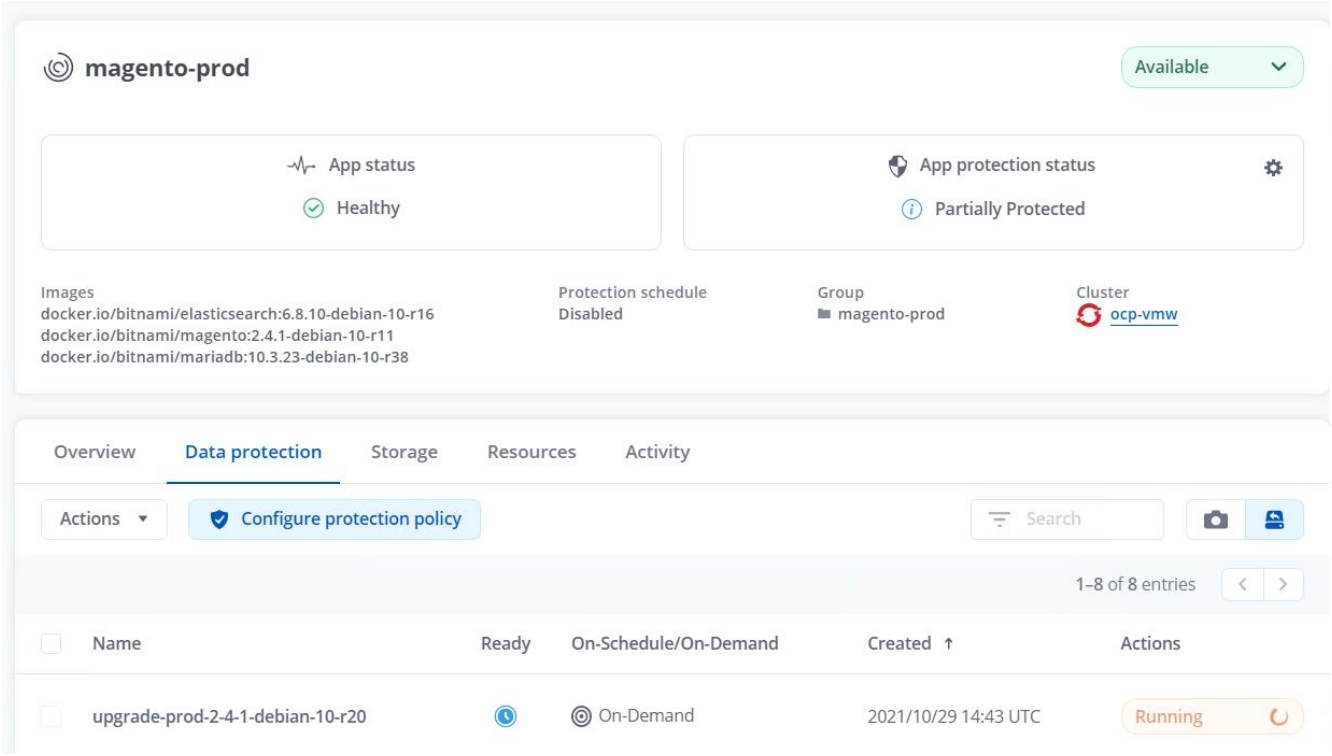


4. Repeat the same procedure for the other two credentials.
5. Go back to the Dashboard, create a pipeline by clicking New Item, and then click Pipeline.
6. Copy the pipeline from the Jenkinsfile here.
7. Paste the pipeline into the Jenkins pipeline section and then click Save.
8. Fill the parameters of the Jenkins pipeline with the respective details including the helm chart version, the Magento application version to be upgraded to, the Astra toolkit version, the Astra Control Center FQDN, the API token, and its instance ID. Specify the docker registry, namespace, and Magento IP of both production and staging environments, and also specify the credential IDs of the credentials created.

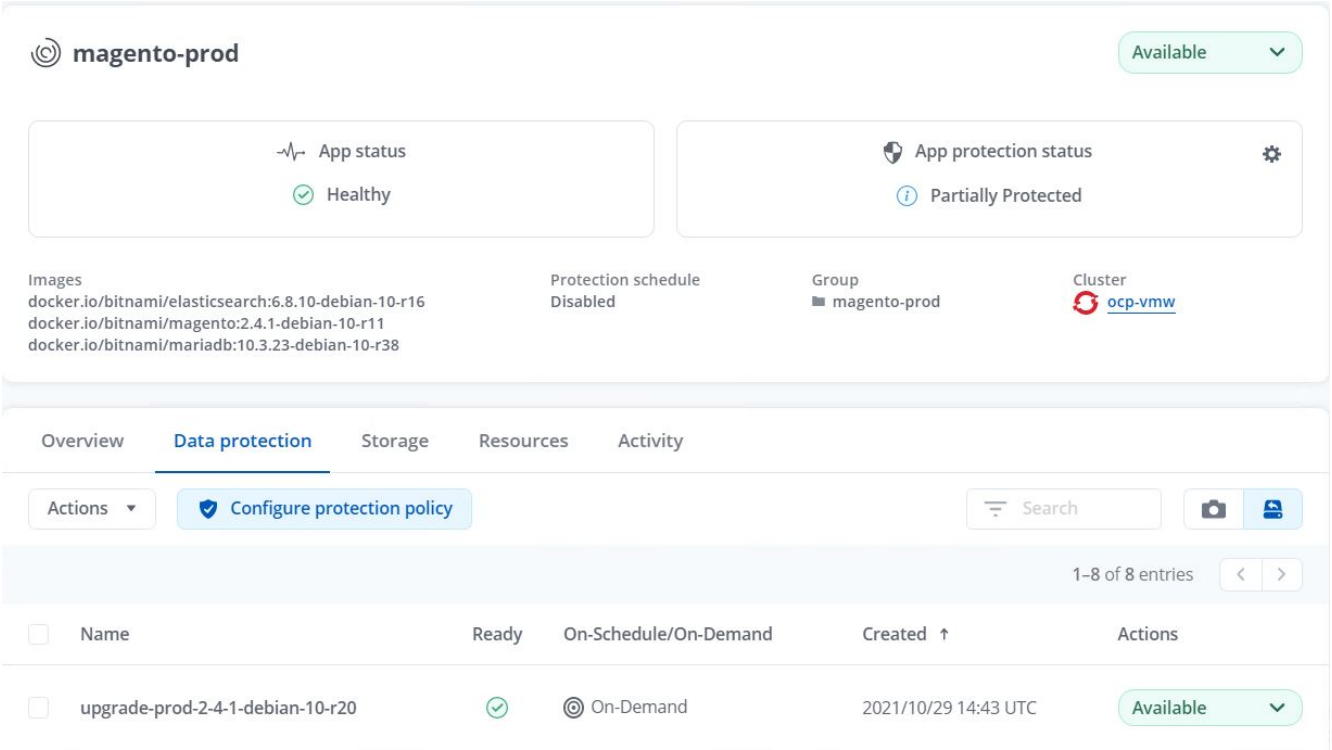
```
MAGENTO_VERSION = '2.4.1-debian-10-r14'
CHART_VERSION = '14'
RELEASE_TYPE = 'MINOR'
ASTRA_TOOLKIT_VERSION = '2.0.2'
ASTRA_API_TOKEN = 'xxxxxxxx'
ASTRA_INSTANCE_ID = 'xxx-xxx-xxx-xxx-xxx'
ASTRA_FQDN = 'netapp-astra-control-center.org.example.com'
DOCKER_REGISTRY = 'docker.io/netapp-solutions-cicd'
PROD_NAMESPACE = 'magento-prod'
PROD_MAGENTO_IP = 'x.x.x.x'
STAGING_NAMESPACE = 'magento-staging'
STAGING_MAGENTO_IP = 'x.x.x.x'
MAGENTO_CREDS = credentials('magento-cred')
MAGENTO_MARIADB_CREDS = credentials('magento-mariadb-cred')
MAGENTO_MARIADB_ROOT_CREDS = credentials('magento-mariadb-root-cred')
```

9. Click Build Now. The pipeline starts executing and progresses through the steps. The application image is first built and uploaded to the container registry.

10. The application backups are initiated via Astra Control.



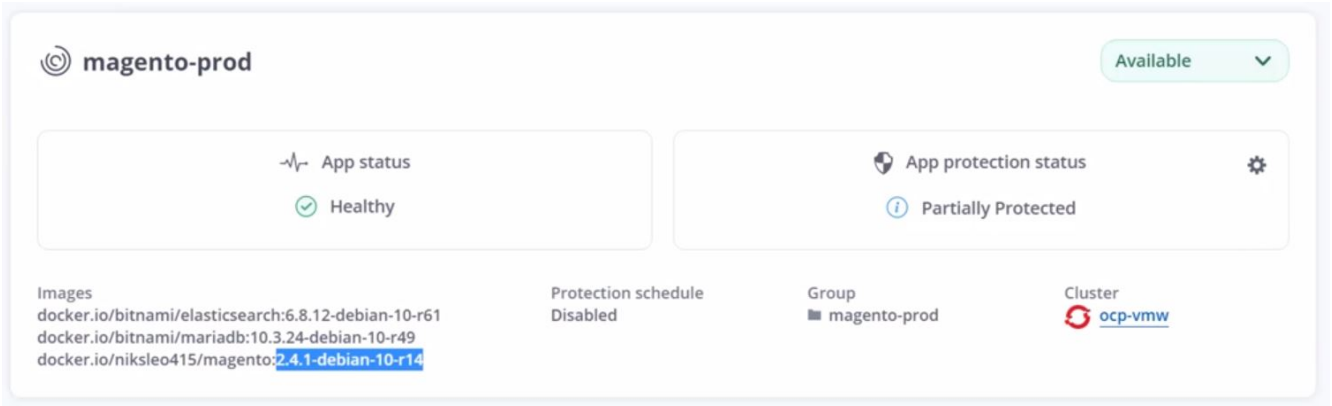
11. After the backup stages have completed successful, verify the backups from the Astra Control Center.



12. The new version of the application is then deployed to the staging environment.

13. After this step is completed, the program waits for the user to approve deployment to production. At this stage, assume that the QA team performs some manual testing and approves production. You can then click Approve to deploy the new version of the application to the production environment.

14. Verify that the production application is also upgraded to the desired version.



As part of the CI/CD pipeline, we demonstrated the ability to protect the application by creating a complete application-aware backup. Because the entire application has been backed up as part of the promotion-to-production pipeline, you can feel more confident about highly automated application deployments. This application-aware backup containing the data, state, and configuration of the application can be useful for numerous DevOps workflows. One important workflow would be to roll back to the previous version of the application in case of unforeseen issues.

Although we demonstrated a CI/CD workflow through with Jenkins tool, the concept can easily and efficiently be extrapolated to different tools and strategies. To see this use case in action, watch the video here.

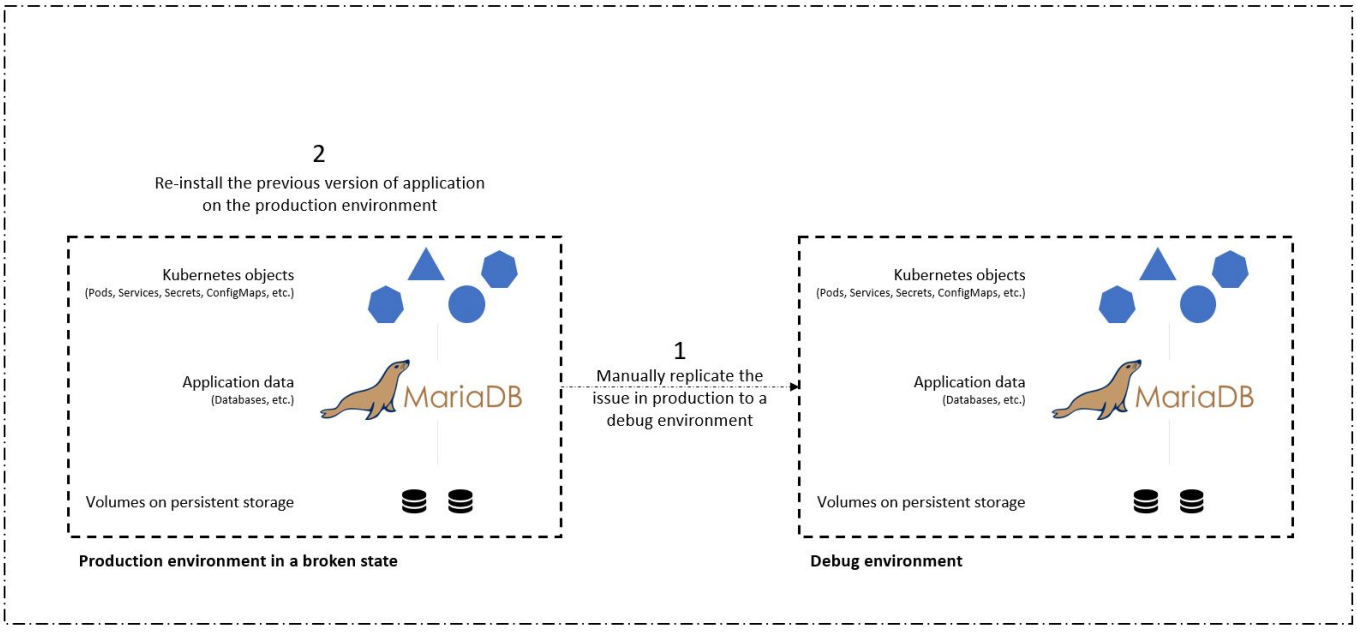
Next: Videos and Demos - DevOps with NetApp Astra.

Use Astra Control to facilitate post-mortem analysis and restore the application

Overview

In the first use case, we demonstrated how to use NetApp Astra Control Center to protect your applications in Kubernetes. That section describes how to integrate application backups via Astra Control directly into your development workflow by using the Python SDK in the NetApp Astra toolkit. This approach allows for the protection of development and production environments by automating on-demand backups during the continuous integration and continuous deployment (CI/CD) process. With this extra layer of application-consistent data protection added to the CI/CD pipeline and the production applications, the development processes is safe if something goes wrong in the process, which promotes good business-continuity practices.

In a traditional workflow, after encountering a failure when the application is upgraded to a new version, the development team would attempt to troubleshoot the issue in real time based on bug reports being provided by customers. Alternatively, at the first sign of trouble, the team could attempt to redeploy the application to a parallel debugging environment to take that process offline. They could redeploy an older code base from a previous version into production, which would restore the application to working order.

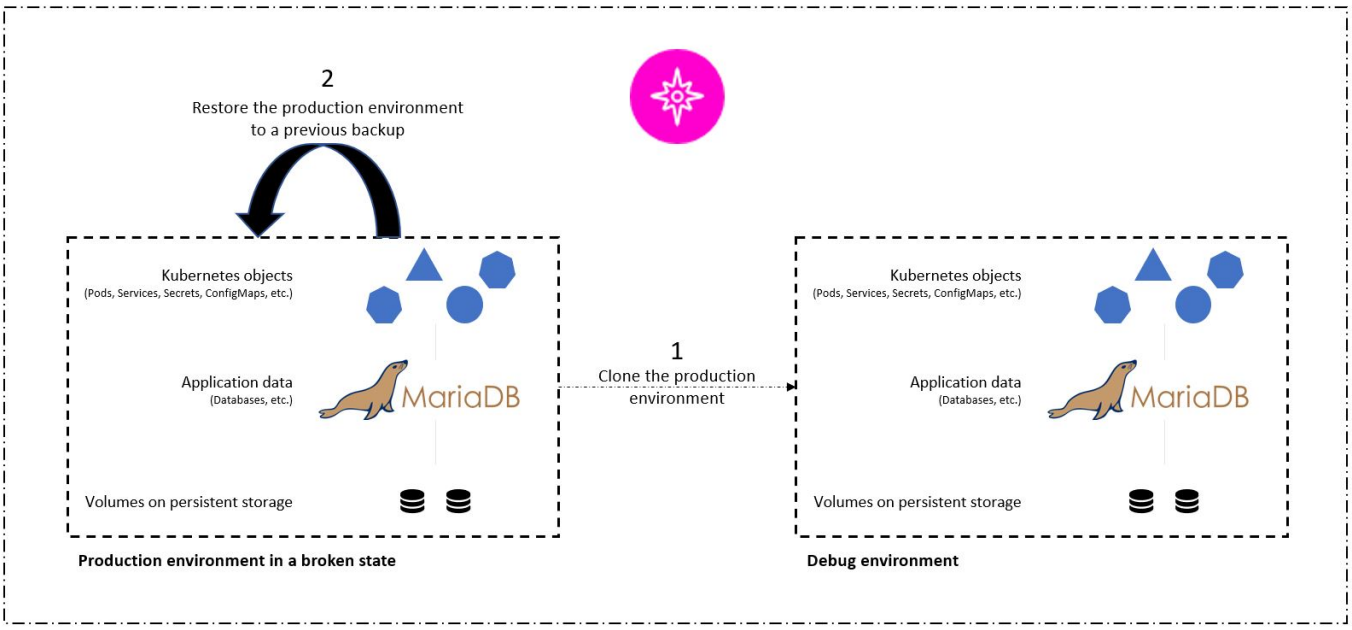


Although this approach works, the team would have to make sure that the state of the broken production app matched that of the version seen in production when the issues occurred. They would also have to spend time promoting the known-good build into production by fetching code from their repository and redeploying the machine images to restore the application to a good running state. Also, in this scenario, we didn't consider whether the production database itself was corrupted by the faulty code. Ideally, there are separate backup processes in place for the database data, but must we assume that they're consistent with the state of the application as it was published? This is where the benefits of stateful and application-consistent backups, restores and clones with Astra Control really show their value.

First, we can use Astra Control to facilitate post-mortem analysis on the state of the application. We do this by

cloning the buggy production version to a parallel testing environment in an application-consistent manner. Having this environment set aside in its bug-ridden state enable us to troubleshoot the problem in real time.

Furthermore, Astra Control supports the in-place restore capability that allows us to restore the production application to a last acceptable backup (that preceded the afflicted version of code). The restored version assumes the position of the previous, buggy production application, in an application-consistent and stateful manner, including the ingress IP previously assigned. As a result, customers accessing the front end would be unaware of the transition to the backup version.



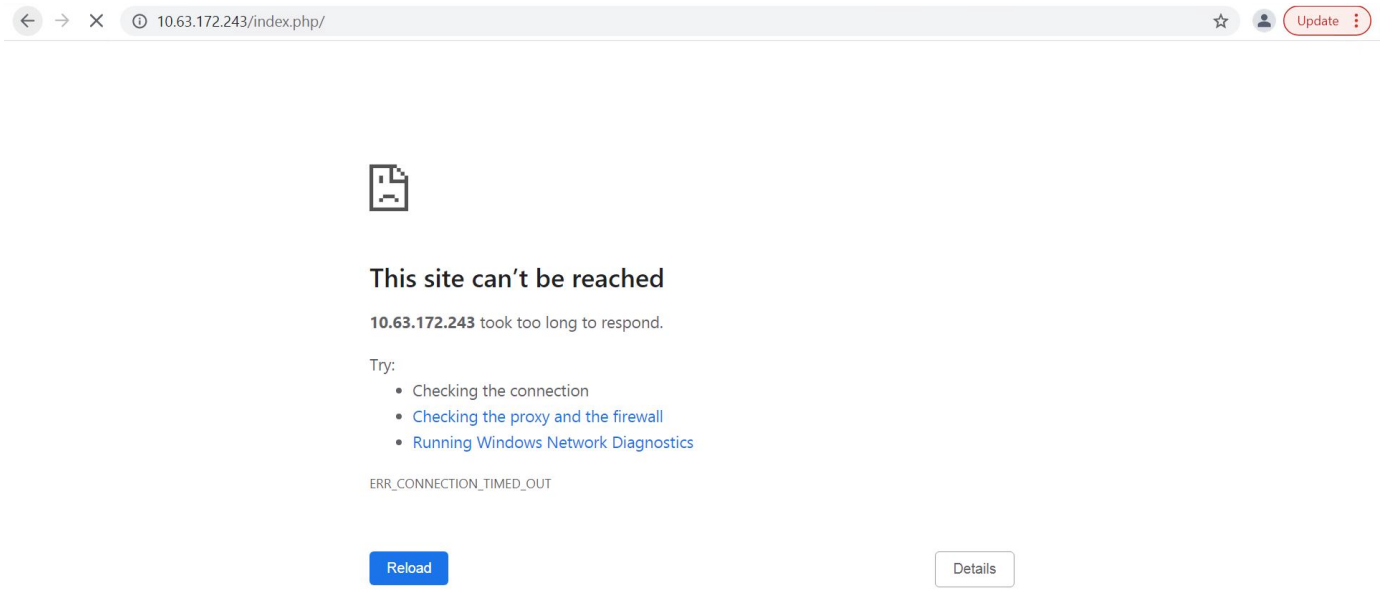
Use-case validation prerequisites

The following tools or platforms were deployed and configured as prerequisites:

- Red Hat OpenShift Container Platform.
- NetApp Astra Trident installed on OpenShift with a backend configured to a NetApp ONTAP system.
- A default storageclass configured, pointing to a NetApp ONTAP backend.
- NetApp Astra Control Center installed on an OpenShift cluster.
- OpenShift cluster added as a managed cluster to Astra Control Center.
- Jenkins installed on an OpenShift cluster.
- Magento application installed in the production environment. The production environment in this use case is a namespace called 'magento-prod' in a Red Hat OpenShift cluster.
- Production application managed by Astra Control Center.
- Known-good backup(s) of the production application captured with Astra Control.

Clone and restore pipeline

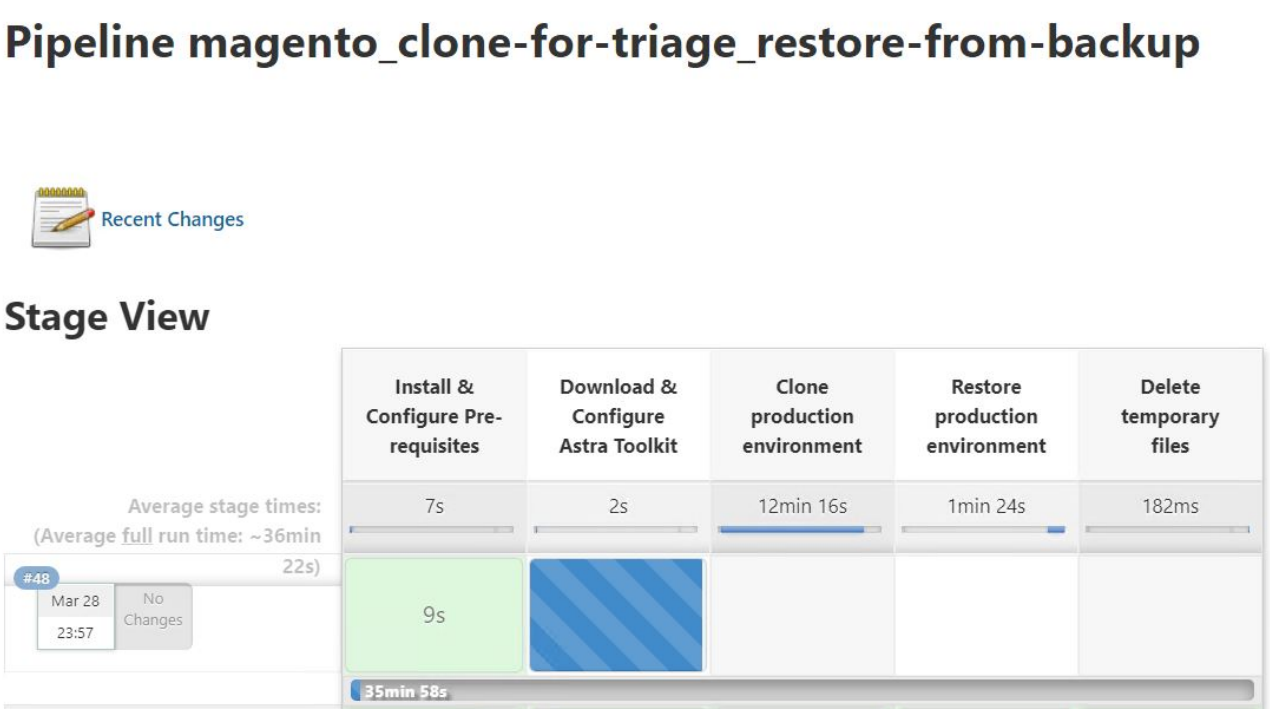
Considering that the application has been upgraded to a new version, the application in the production environment (`magento-prod`) isn't behaving as intended after the upgrade. Let's assume that the data being returned by front-end queries doesn't match the request or that the database has in fact been corrupted. To clone and restore the pipeline, complete the following steps:



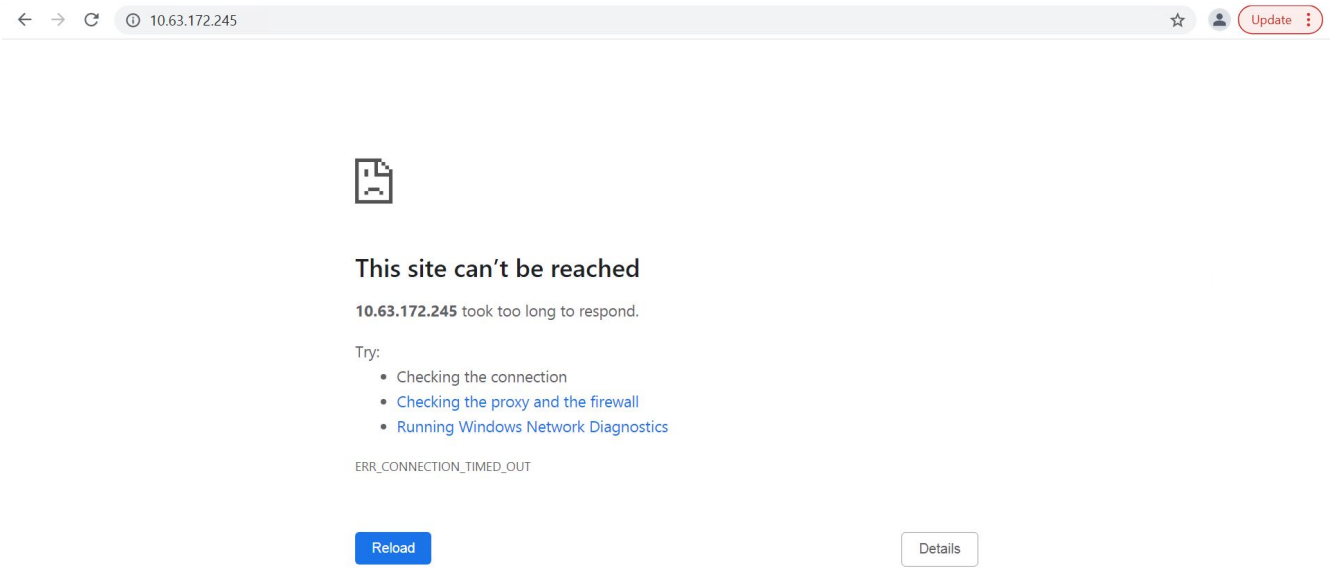
1. Log into Jenkins and create a pipeline by clicking New Item and then Pipeline.
2. Copy the pipeline from the Jenkinsfile here.
3. Paste the pipeline into the Jenkins pipeline section and then click Save.
4. Fill the parameters of the Jenkins pipeline with the respective details like the current Magento application version in production, the Astra Control Center FQDN, the API token, the instance ID and application name or namespace of production and debug environments, and the source and destination cluster names. For the purpose of this use case, the production environment is a namespace called 'magento-prod' and the debug environment is a namespace called 'magento-debug' configured on a Red Hat OpenShift cluster.

```
MAGENTO_VERSION = '2.4.1-debian-10-r14'
ASTRA_TOOLKIT_VERSION = '2.0.2'
ASTRA_API_TOKEN = 'xxxxx'
ASTRA_INSTANCE_ID = 'xxx-xxx-xxx-xxx-xxx'
ASTRA_FQDN = 'netapp-astra-control-center.org.example.com'
PROD_APP_NAME = 'magento-prod'
DEBUG_APP_NAME = 'magento-debug'
DEBUG_NAMESPACE = 'magento-debug'
PROD_KUBERNETES_CLUSTER = 'ocp-vmw'
DEBUG_KUBERNETES_CLUSTER = 'ocp-vmw'
```

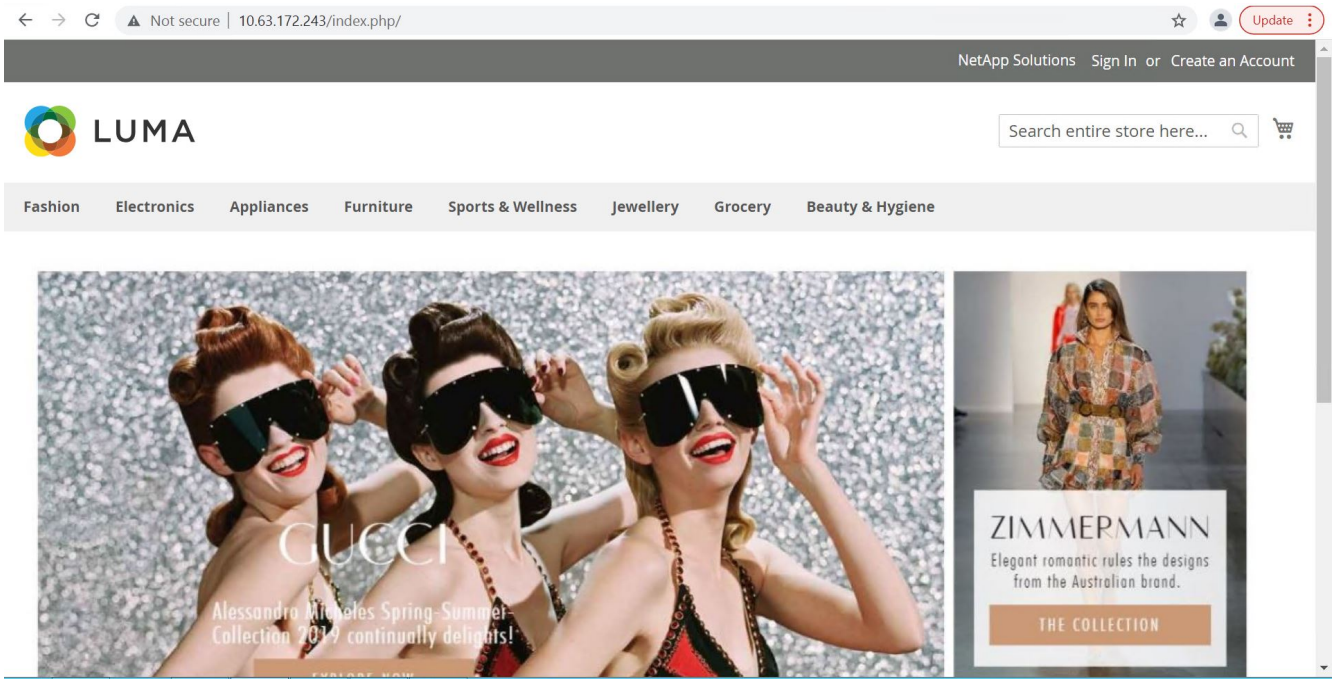
5. Click Build Now. The pipeline starts executing and progresses through the steps. The application is first cloned in the current state to a debug environment, and the application is then restored to the known-working backup.



6. Verify that the cloned application is the bug-containing version.



7. Verify that the production environment is restored to a working backup, and the application in production works as expected.



These two operations in tandem expedite the return to normal business operations. To see this use case in action, watch the video here.

Next: Videos and Demos - DevOps with NetApp Astra.

Accelerating software development with NetApp FlexClone technology

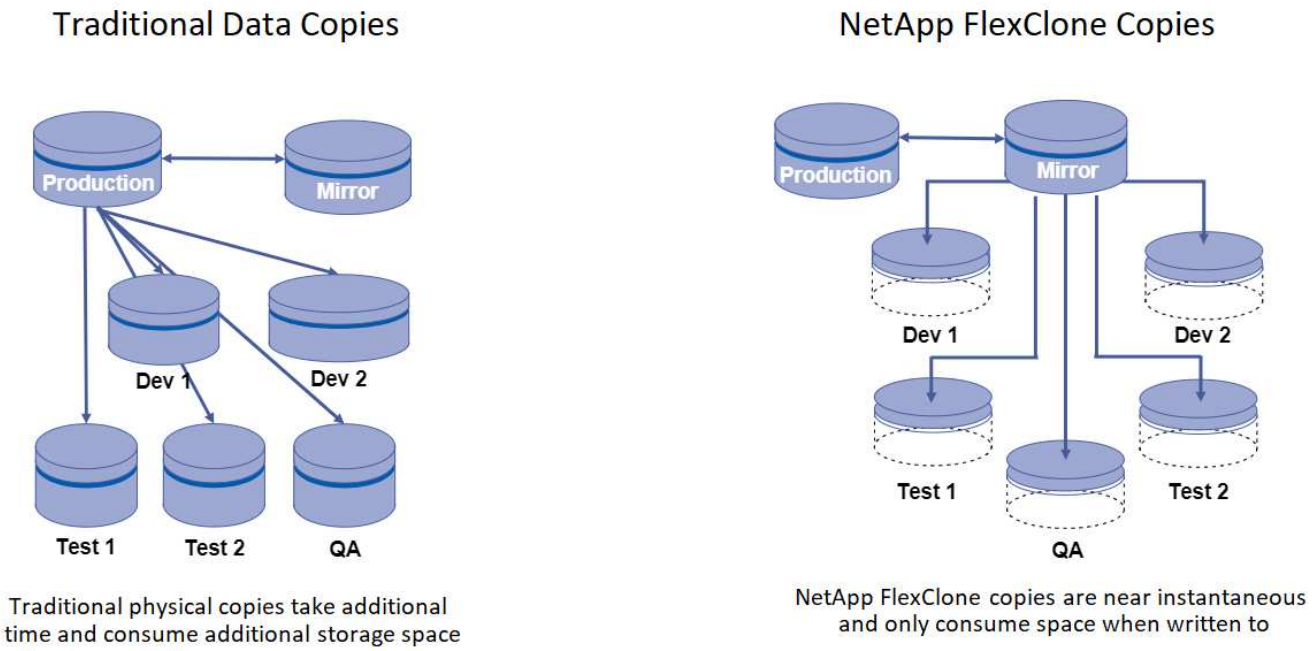
Overview

Cloning a deployed application in a Kubernetes cluster is a very useful tool for developers that would like to expedite their workflows by sharing environments with partners or by testing new versions of code in a development environment without interfering with the version they are currently working on. The stateful and application-consistent cloning of a Kubernetes application is a major feature included with NetApp Astra Control, alongside the backup and restore of applications. As a bonus, if an application is cloned within the same Kubernetes cluster using the same storage backend, Astra Control defaults to using NetApp FlexClone technology for the duplication of persistent data volumes, speeding up the process significantly. By accelerating this process, the cloned environment is provisioned and available for use in a few moments, allowing developers to resume their work with just a brief pause when compared to redeploying their test or development environment. As an additional convenience, all of the functions available in NetApp Astra Control can be called with an API, which allows for easy integration into automation frameworks like Ansible. Therefore, environments can be staged even more rapidly because only minor changes are needed in a playbook or role to begin the cloning procedure.

What is NetApp FlexClone technology?

NetApp FlexClone technology is a writeable, point-in-time snapshot-based copy of a NetApp FlexVol. They are provisioned almost instantly, contain all of the data from the source volume, and consume no additional storage space until the data in the new volume begins to diverge from the source. They are often used in development or template-based environments when multiple copies of data are useful for staging purposes and storage systems have limited resources for provisioning these volumes. Compared to a traditional storage system in which data must be copied multiple times resulting in the consumption of significant storage space and time,

NetApp FlexClone technology accelerates storage-dependant tasks.



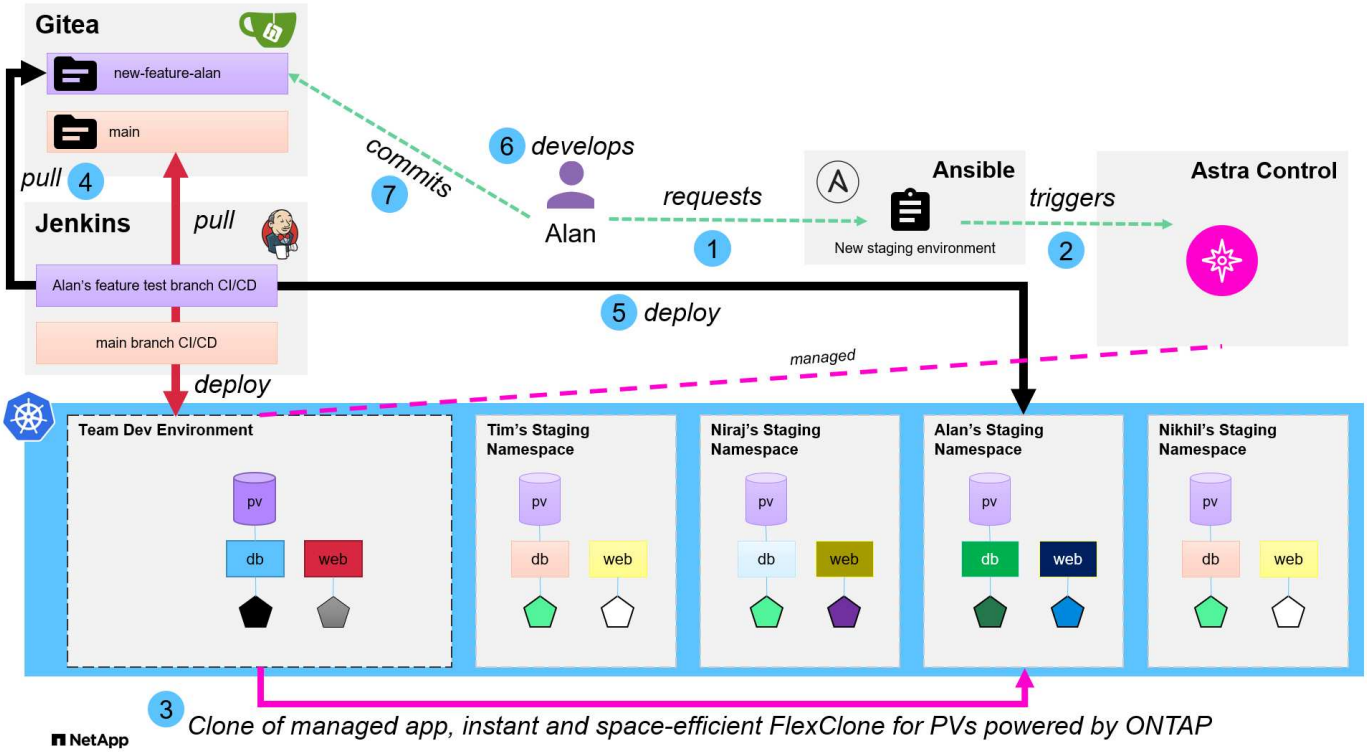
To find out more about NetApp FlexClone technology, visit the page on NetApp Docs.

Prerequisites

1. A supported Kubernetes Distribution, such as Red Hat OpenShift 4.6.8+, Rancher 2.5+, or Kubernetes 1.19+.
2. NetApp Astra Control Center 21.12+.
3. A NetApp ONTAP system with a storage backend configured through NetApp Astra Trident.
4. Ansible 2.9+.
5. Templates for the environments that you'd like to clone as managed applications in NetApp Astra Control.

Use-case introduction

For this use case, we visualize something similar to the following workflow:



1. A user runs the ansible playbook to create a new staging environment.
2. Ansible uses the URI-API module to call out to Astra Control to execute the cloning operation.
3. Astra Control executes a cloning operation on a preprovisioned template environment, thus creating a new managed application.

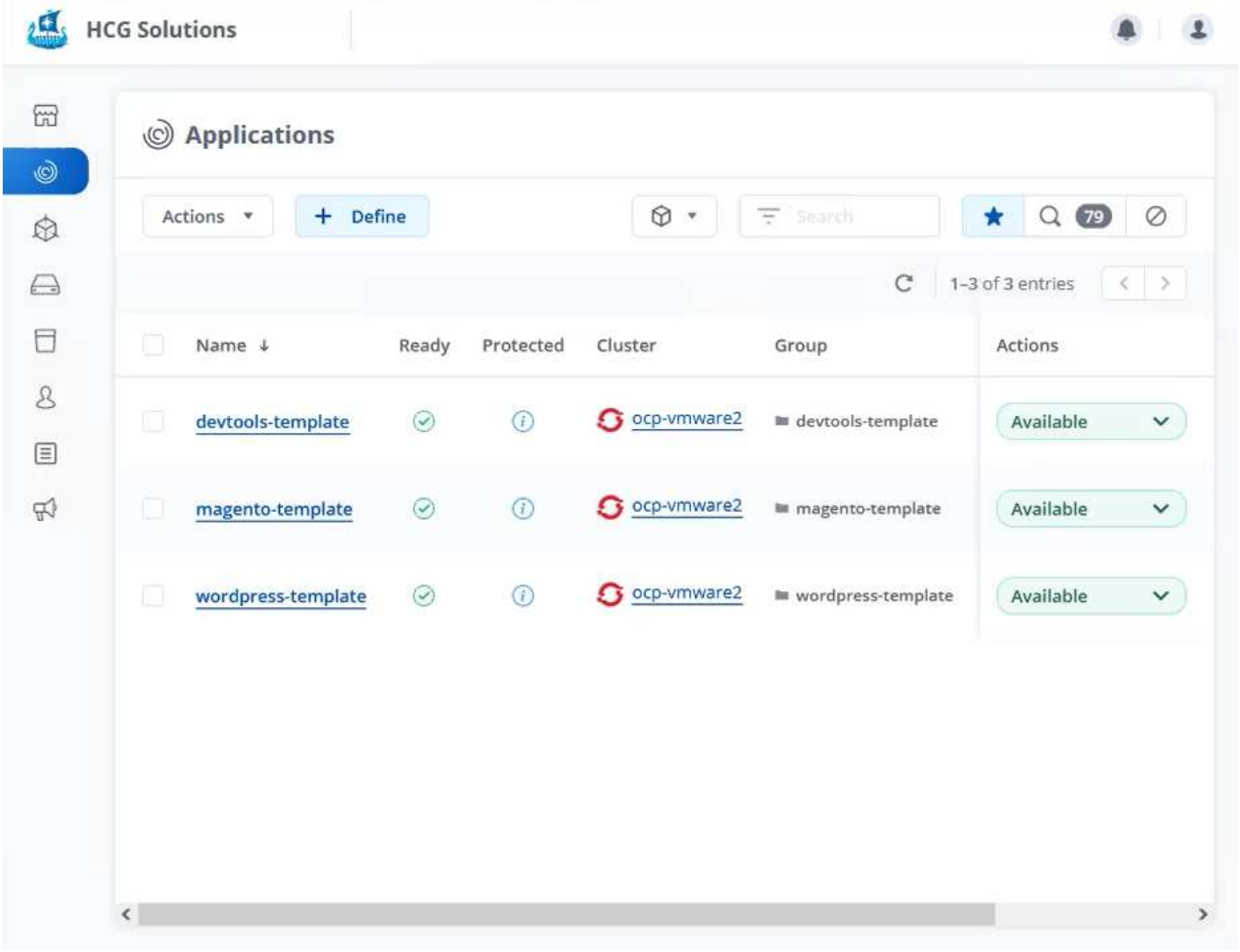
> This environment can be a single standalone application in development or an entire development environment like a Jenkins CI/CD pipeline.

4. The user then pulls a version of their code into the cloned dev environment from an online repository like Gitea.
5. The new version of the application is deployed and managed by NetApp Astra Control.

> Both of these processes can be automated.

6. The user can develop new code in this cloned environment.
7. When the user is satisfied with their development efforts, they can push the code back to the hosted repository.

The use case presented here depends on the existence of golden templates for the particular environments or applications you would like to clone. In our environment we have created three such templates, one for a Wordpress deployment, one for a Magento deployment, and one for a Jenkins CI/CD environment with Gitea that we have titled DevTools.



Each of these environments is managed by NetApp Astra control, with persistent volumes currently stored on a NetApp ONTAP storage system with an NFS backend provided by NetApp Astra Trident.

Use-case validation

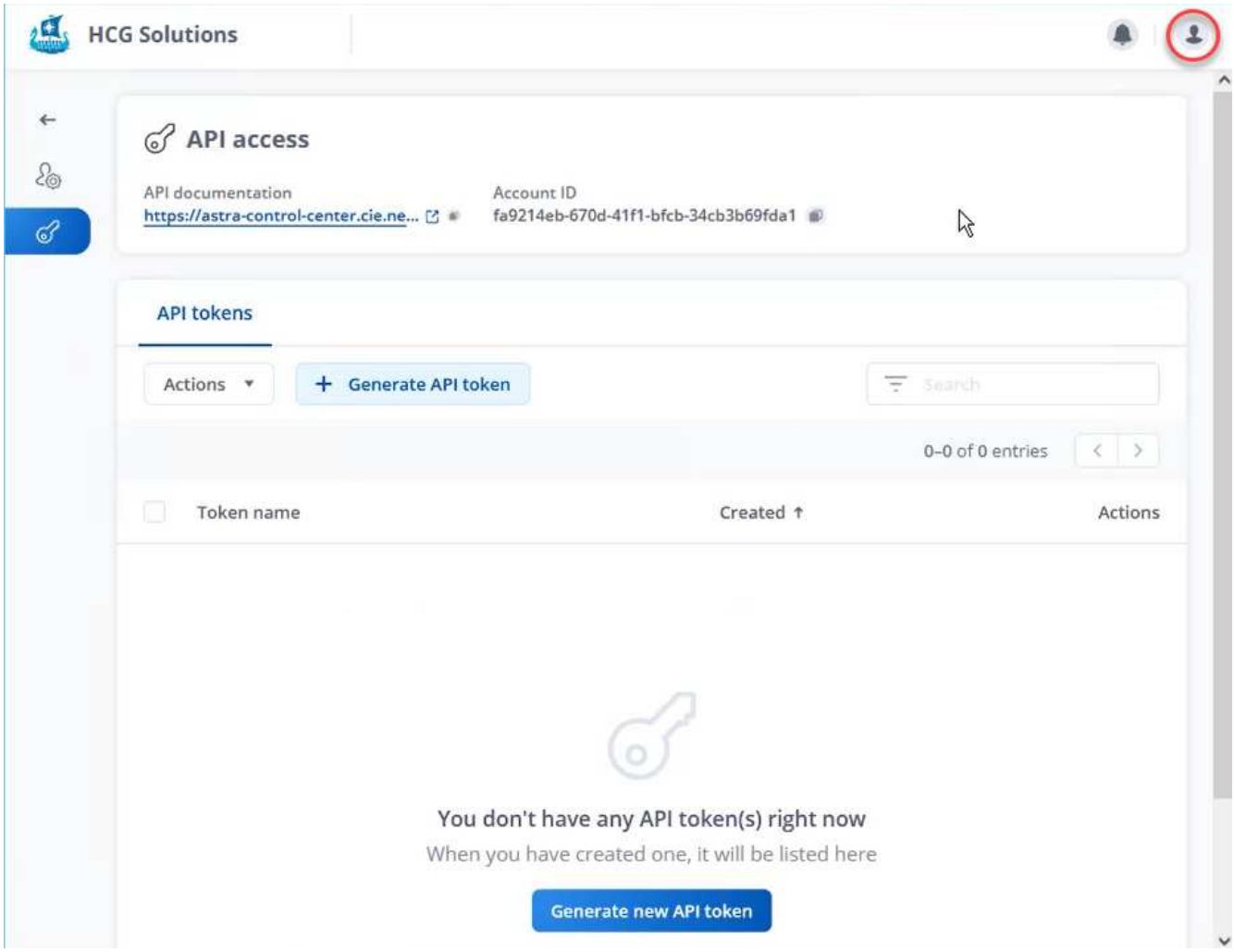
1. Clone the ansible toolkit provided by the NetApp Solutions Engineering team, which includes the cloning role and the application update playbook.

```
[netapp-user@rhel7 ~]$ git clone https://github.com/NetApp-
Automation/na_astra_control_suite.git
[netapp-user@rhel7 ~]$ cd na_astra_control_suite
```

2. Edit `vars/clone_vars.yml` and fill in the global values that fit your Astra Control environment.

```
astra_control_fqdn: astra-control-center.example.com
astra_control_account_id: "xxxx-xxxx-xxxx-xxxx-xxxx"
astra_control_api_token: "xxxxx"
```

The global environment values you need to fill out are available under the user profile icon in NetApp Astra Control under the API Access menu.



3. With the global variables completed, you can choose the values for the specific application you wish to clone. To clone the devtools environment to a personal environment called `alan-devtools`, you would do the following:

```
clone_details:
  - clone_name: alan-devtools
    destination_namespace: alan-dev-namespace
    source_cluster_name: ocp-vmware2
    destination_cluster_name: ocp-vmware2
    source_application_name: devtools-template
```

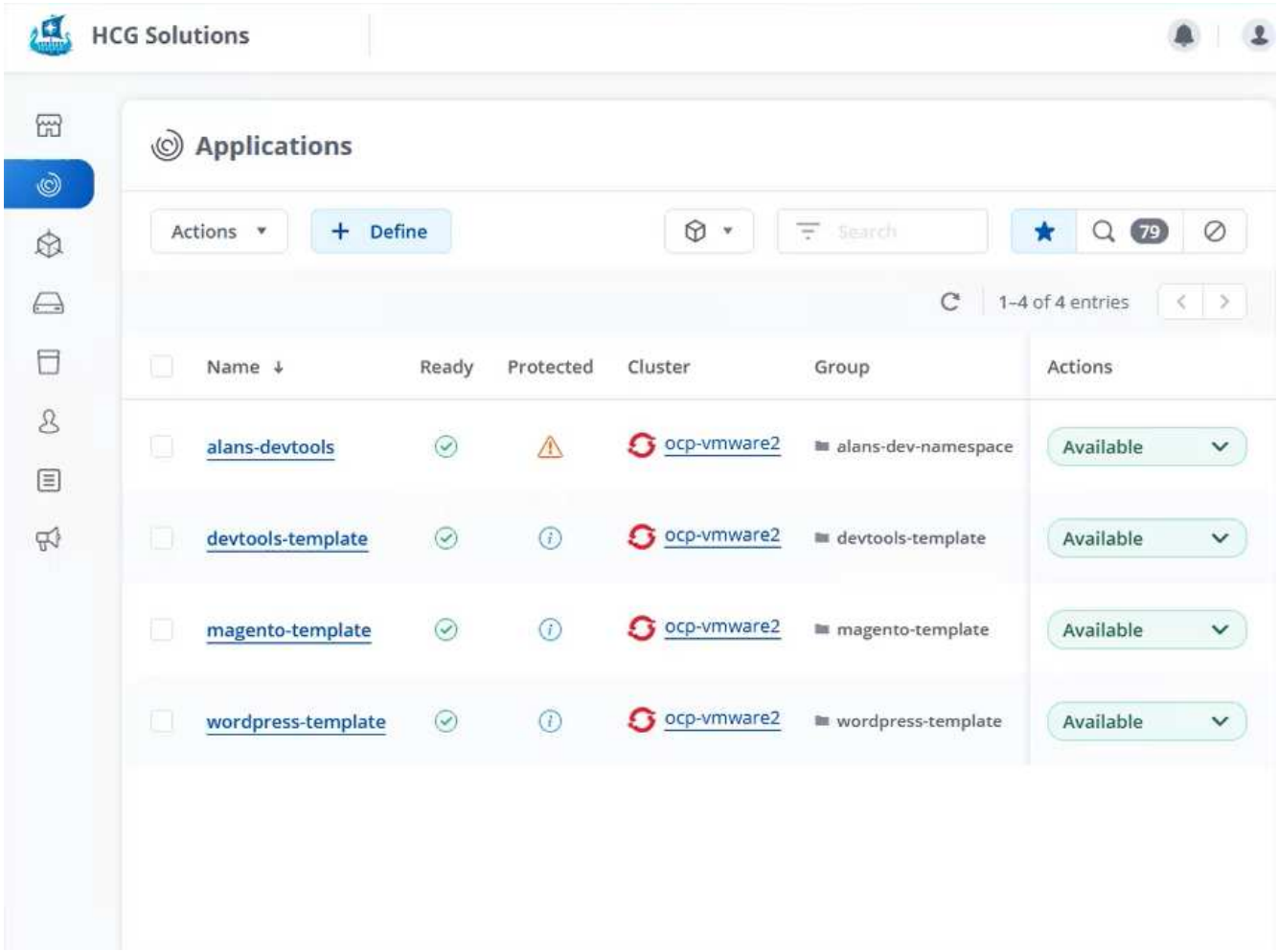
To take advantage of NetApp FlexClone technology in the cloning process, `src-cluster` and `dest-cluster` must be the same.

4. You can now execute the playbook to clone the application.

```
[netapp-user@rhel7 na_astra_control_suite]$ ansible-playbook -K
clone_app_playbook.yml]
```

The playbook as written must be run by the root user or someone that can escalate through the sudo process by passing the "-K" argument.

5. When the playbook completes its run, the cloned application shows as available in the Astra Control Center console.



6. A user can then log into the Kubernetes environment where the application was deployed, verify that the application is exposed with a new IP address, and start their development work.

For a demonstration of this use case and a example of upgrading an application, see here.

Next: Videos and Demos - DevOps with NetApp Astra.

Copyright information

Trademark information