DevOps on Zerostack

Devops philosophy and best practices

Zerostack sees devops as a business best practice that helps organizations break down siloed environments, increase the velocity of software development, and speed product delivery. Standardized tools, infrastructure, and automation are the keys to a successful digital transformation within an organization. Standardization is not enough, DevOps requires a cultural, and a technological, shift within an organization to be properly implemented. The term digital transformation, in part, refers to the cultural shift needed to blend development and operations teams together, then implement automated processes.

In a modern organization, devops can be seen as a set of tools and business processes that allow separate teams to function as a single unit with a single workflow. DevOps pulls in some of the core concepts of agile software development. However, unlike the agile software model, DevOps focuses on all parts of the development cycle from the initial concept, to delivery of the end product.

Zerostack sees DevOps as a way for siloed teams, who may not have interacted in the past, to interact with each other based on their core competencies. The end result of these interactions should be automated business processes that enable rapid and successful product delivery. Pulling these teams together will require a cultural transition, internal champions, robust infrastructure, and the tools necessary to facilitate an organization's digital transformation.

DevOps Culture

The key to a successful DevOps cultural shift is communication. All of the teams involved in the design, build, and deployment of a product must be on the same page. In order for these teams to get on, and stay on, the same page a common set of tools and processes must be agreed upon and put in place. All of the teams involved in the DevOps transition should have a common set of goals and be accountable to each other, and should act like a single team.

All teams involved in DevOps should also relentlessly pursue process improvement. Process improvement can involve automation, streamlining code, and cutting out excess steps in the pipeline. New tool sets, and flexible infrastructure, are essential to process improvement. A modern, flexible, infrastructure allows your organization to quickly respond to organizational and process change. In some larger organizations there are a mass of legacy tools, processes, and infrastructure already in play. Zerostack can help mitigate the disruption that can occur during a DevOps transition, since current infrastructure can be used. Once Zerostack is implemented,

legacy apps can be transitioned to a cutting edge, flexible infrastructure and can live next to modern DevOps and cloud native applications.

DevOps Components

Devops Buildout

Zerostack furnishes a script that deploys the skeleton of a best practices DevOps infrastructure on top of Zerostack. In order to deploy the DevOps environment, download the script, source your RC file, and run the script.

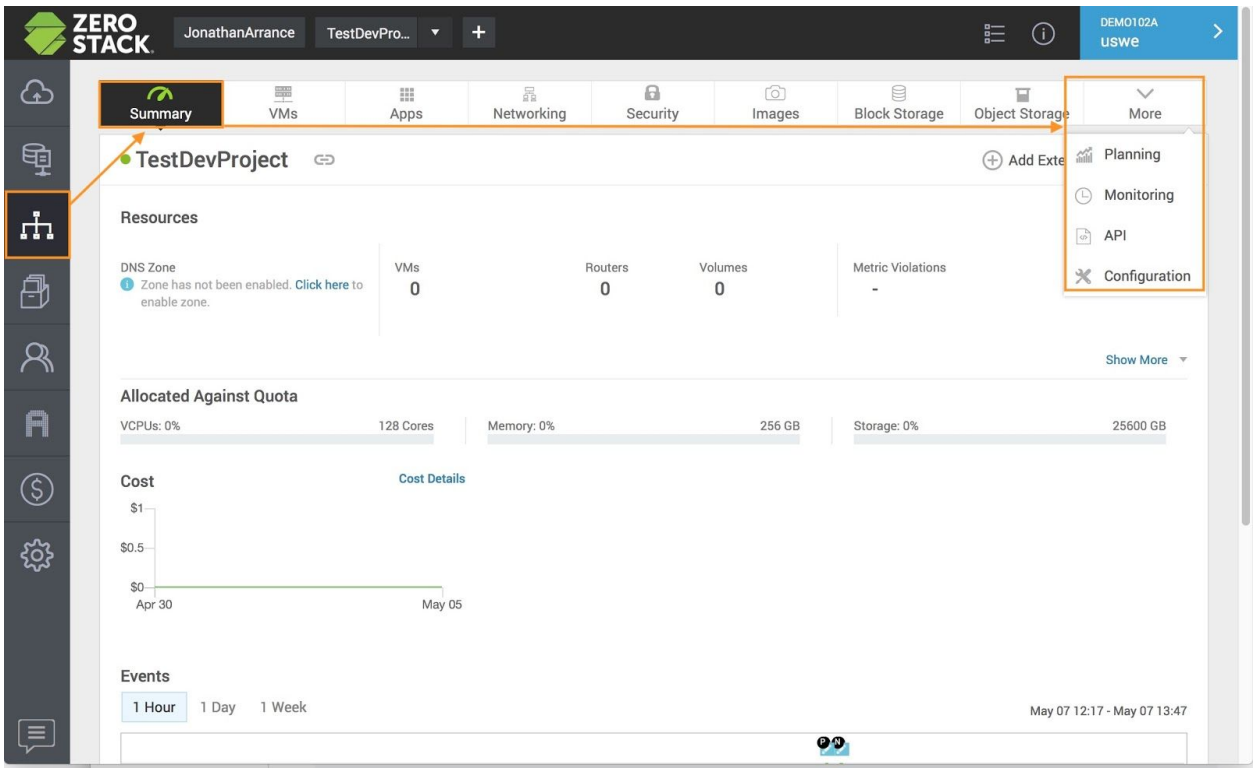
Download the DevOps deploy.py script to your personal system.

```
# git clone https://github.com/Zerostack-open/zs-devops-demo.git
```

Log into the Zerostack platform and pull down your RC file and Key File

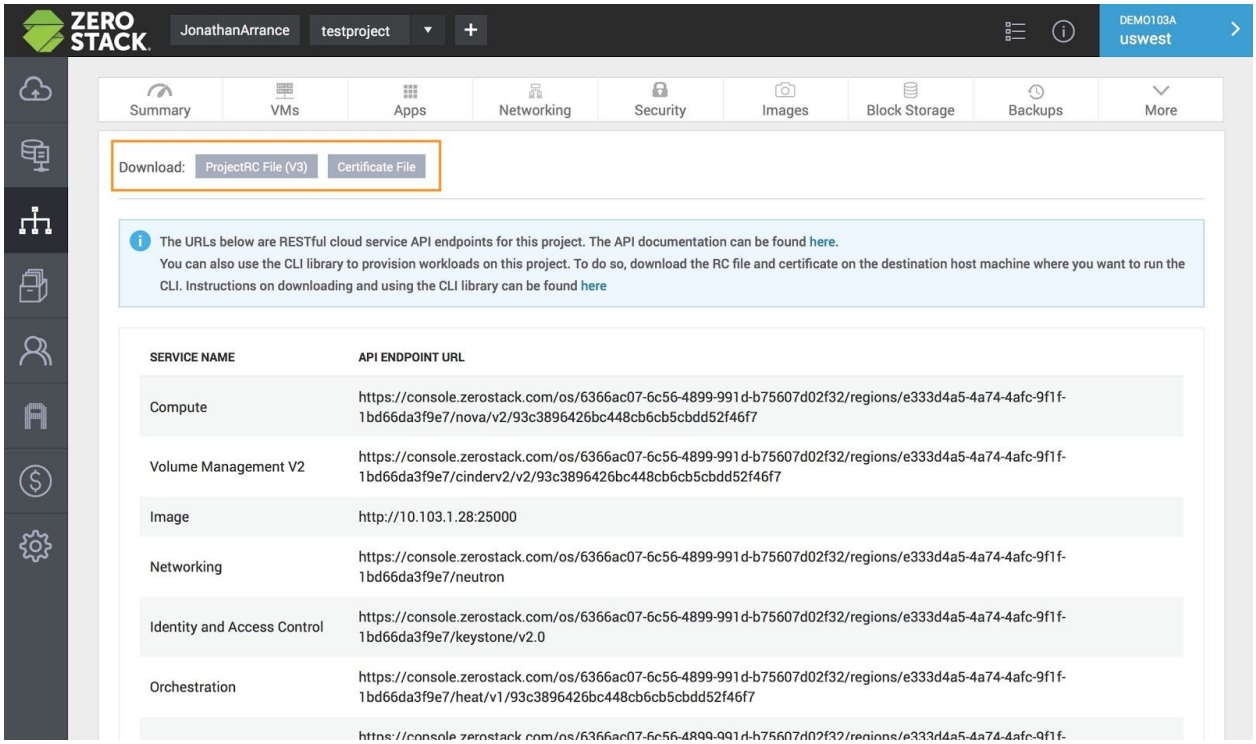
URL: https://console.zerostack.com

Once in your project, click on *More*, and the on the API icon.



Once in the API section you will be able to pull down the RC file and the Key file.

NOTE: The script will fail if you do not source the RC file, the login credentials and authentication endpoint are needed in order to build the environment.



Once the files are downloaded, your RC file should look like the following example.

```
USER_NAME=myusername
USER_PASSWORD=mypassword
USER_DOMAIN=myBU - or admin.local for cloud admin
PROJECT_DOMAIN=myBU
USER_REGION=uswe - NOTE: This needs to be added.
USER_PROJECT=myproject
ZS_CERT_FILE=~/zs_Certificate_ca.bundle - fully qualified path to cert
export
OS_AUTH_URL=https://console.zerostack.com/os/6366ac07-6c56-4899-991d-b75607d02f
32/regions/e333d4a5-4a74-4afc-9f1f-1bd66da3f9e7/keystone/v3
export OS_CACERT=$ZS_CERT_FILE
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=1
export OS_VOLUME_API_VERSION=2
export OS_USERNAME=$USER_NAME
export OS_USER_DOMAIN_NAME=$USER_DOMAIN
export OS_PASSWORD=$USER_PASSWORD
export OS_PROJECT_NAME=$USER_PROJECT
export OS_PROJECT_DOMAIN_NAME=$PROJECT_DOMAIN
```

Once the RC file is set, it will need to be sourced.

```
# source ~/path/to/rc/file/zs_rc.txt
```

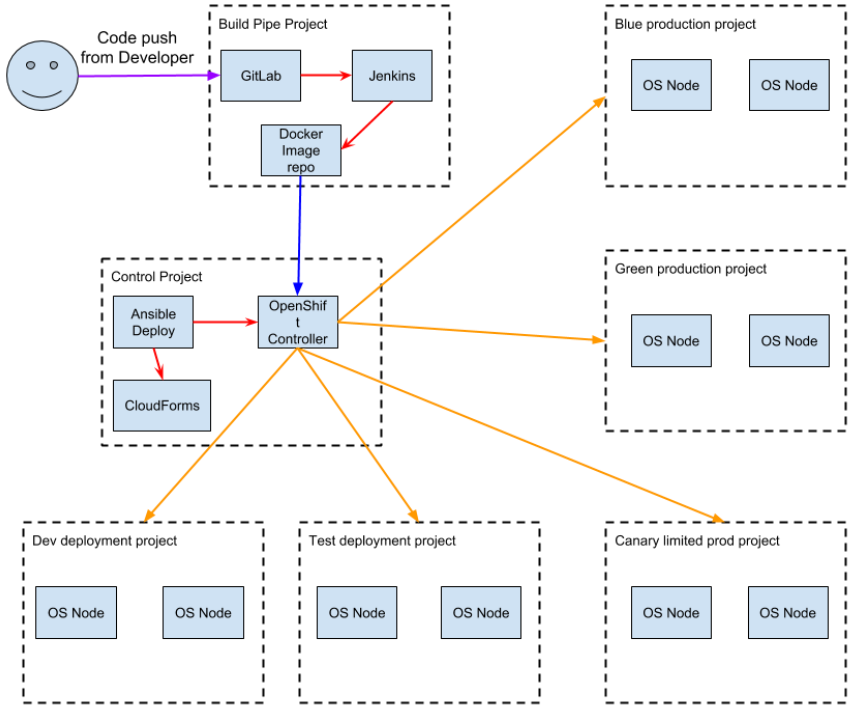
Now, run the DevOps deploy.py script

```
# python deploy.py
```

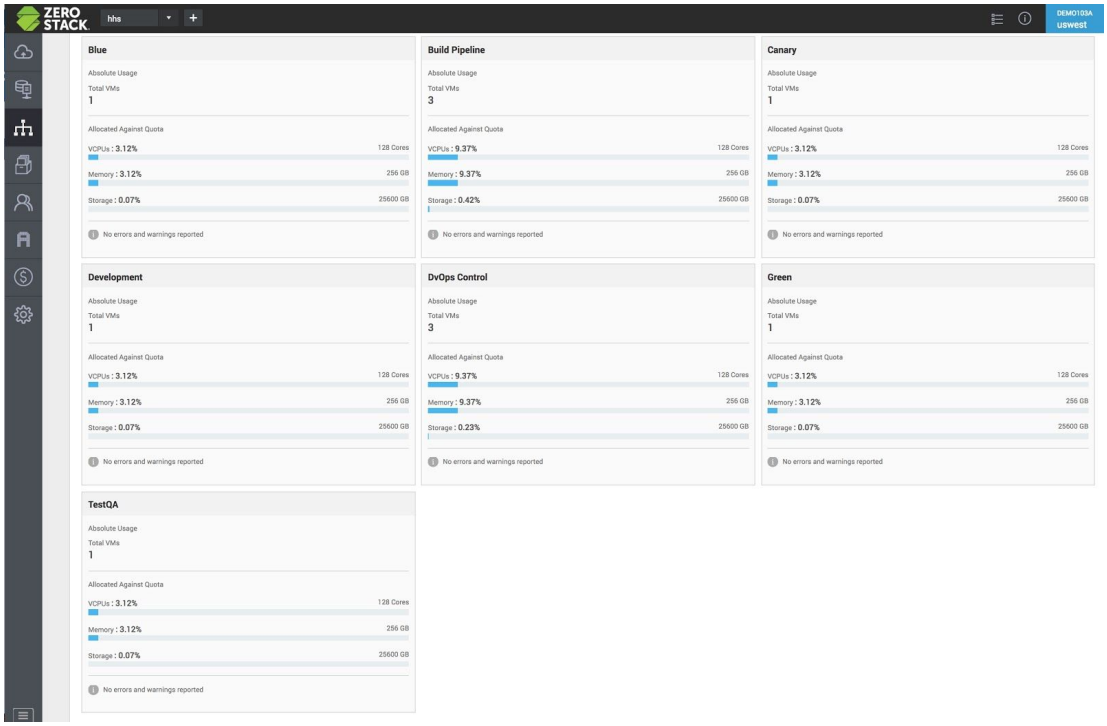
What gets deployed?

The deploy.py script will deploy the following environment. All of the VMs and apps will be connected via a virtual DevOps network that is created with the control project. All of the VMs will use the same ssh keypair for access.

NOTE: The script does not do any configuration, tools such as GitLab and Jenkins will need to be configured after they are deployed. Please see the following sections.



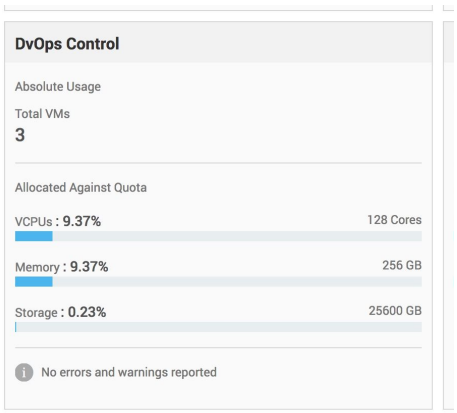
If this layout does not suit your needs, it can be very easily modified in the script. The script is flat a procedural and can be used as a guide for building other automated tasks.



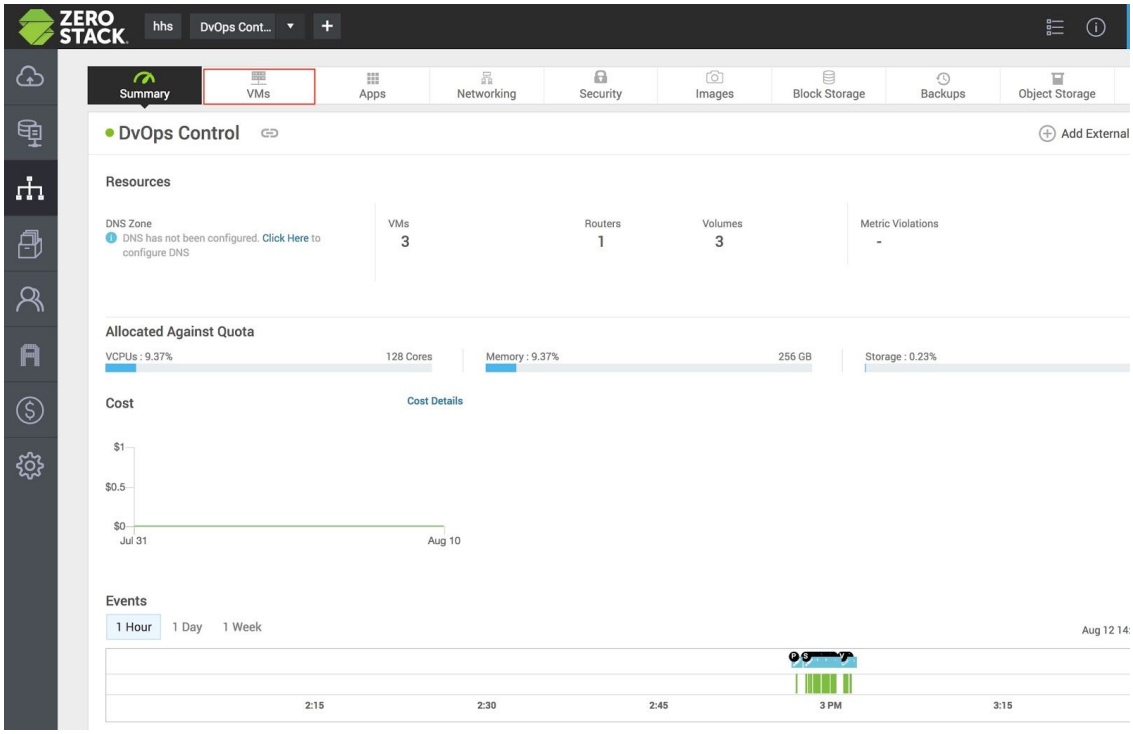
DevOps Control

The Devops control project, is the center of the DevOps deployment. When the control project is built within the BU, a common DevOps network, and security keys are created. Also skeleton VMs for Ansible, Cloudforms, and Openshift will be built. If these VMs are not needed, delete them to reclaim resources.

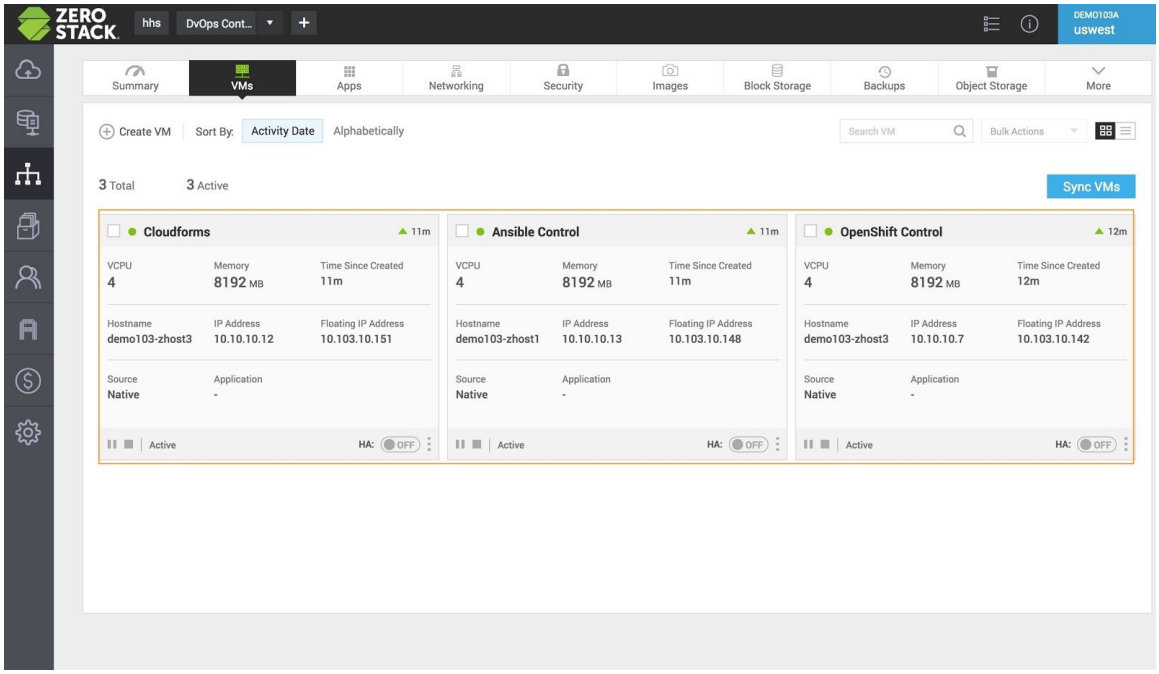
Look for the DevOps Control project in the BU.



Click on the project tile, once the project interface comes up, click on VMS.

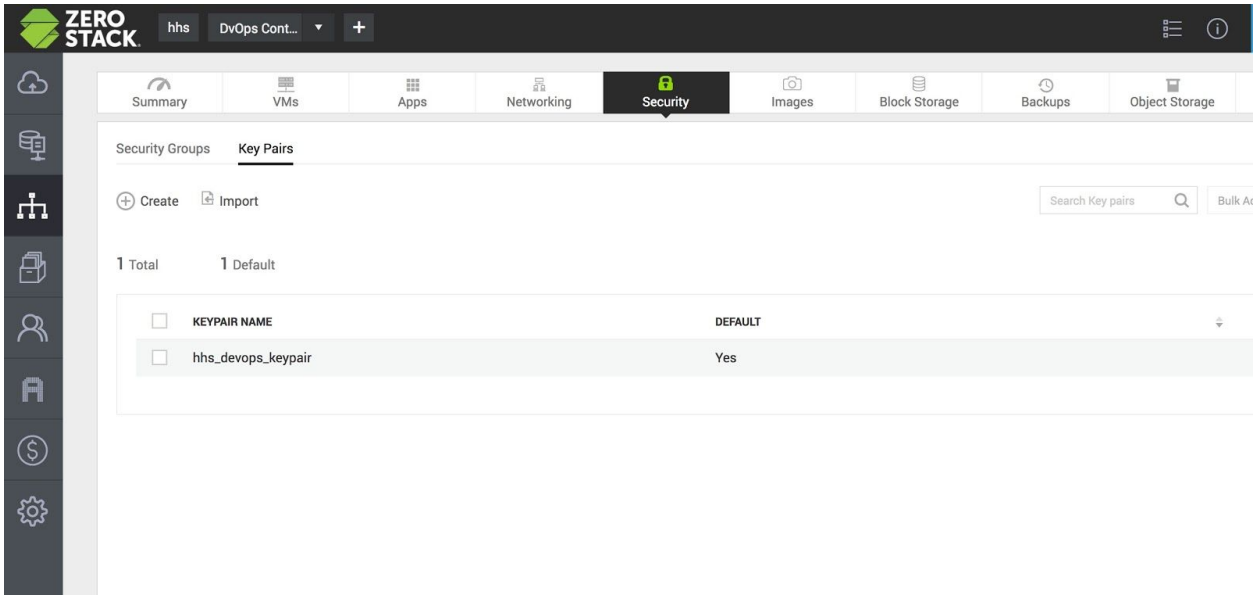


Once in the VMs interface you will see the control VMs that were spun up when the devops deploy script was run.



The Ansible VM will be set up with a base deployment of Ansible and will need to have playbooks uploaded to it. The CloudForms VM will have the upstream ManageIQ code loaded into a Docker container. If RedHat Cloudforms is needed the deployment scripts are attached in the appendix and can be modified. The OpenShift VM is clean and will need to have OpenShift deployed to it.

In order to login to the VM and install the software, the public key will need to be downloaded.



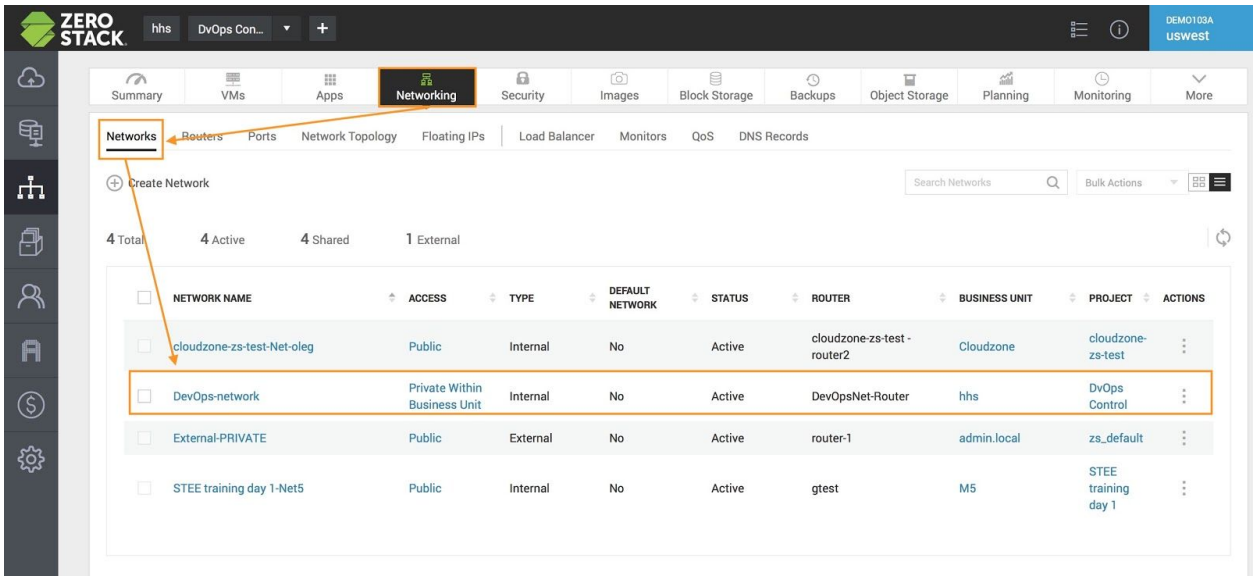
Once the key is downloaded, use it to ssh to the VM.

NOTE: This example is on a *NIX based system.
NOTE: In this deploy script CentOS 7 is used, the username is centos.

```
# chmod 0400 "keypairname"
# ssh -i "keypairname" username@ipaddress.com
```

The devops keypair is the default key pair used for all VMs and apps deployed with the deploy.py script.

A common devops network is also created within the DevOps control project. The DevOps network is used to connect all of the VMs and Apps in the various projects.



DevOps Pipeline Project

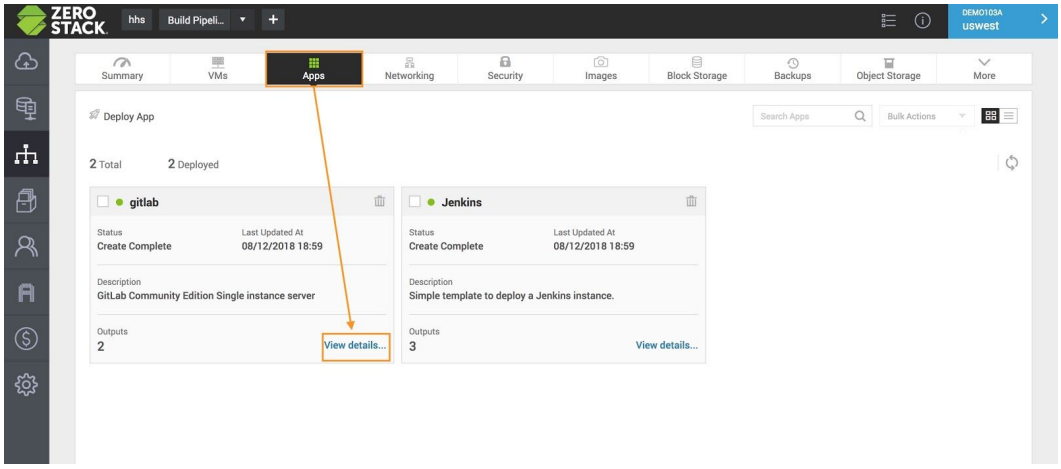
Once the DevOps pipeline project is deployed, the Jenkins and GitLab app will need to be configured.

GitLab

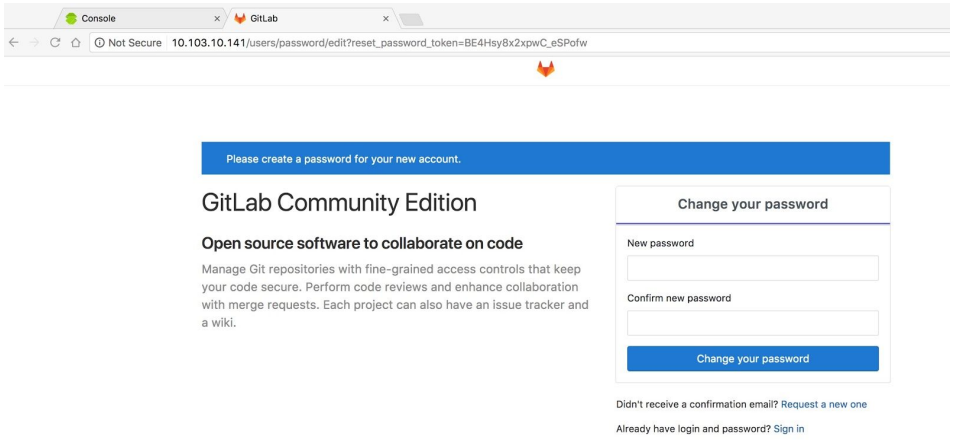
Gitlab will be used as the code repository in our example environment. Devops deploy automatically deploys the GitLab App stack from the Zerostack app repository. Once the app stack is deployed, Gitlab will need to be configured.

Configure Gitlab

In order to get to the GitLab interface, click on the view details icon. Once in the appstack view, use the ip address assigned.

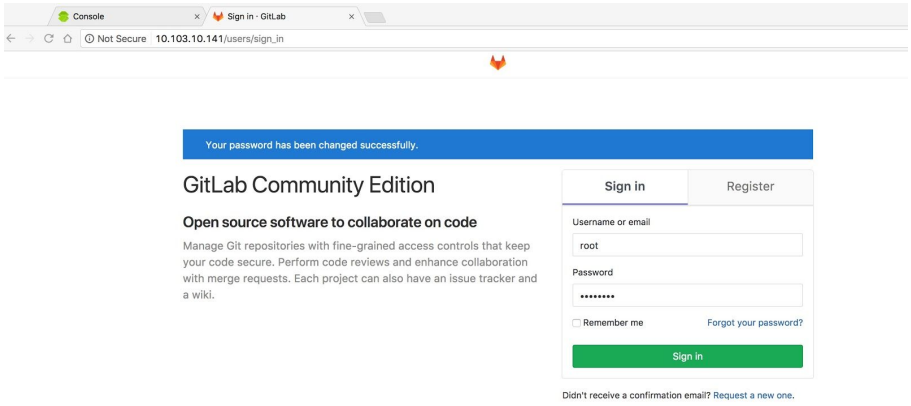


Once the GitLab interface comes up, a root password will need to be set.

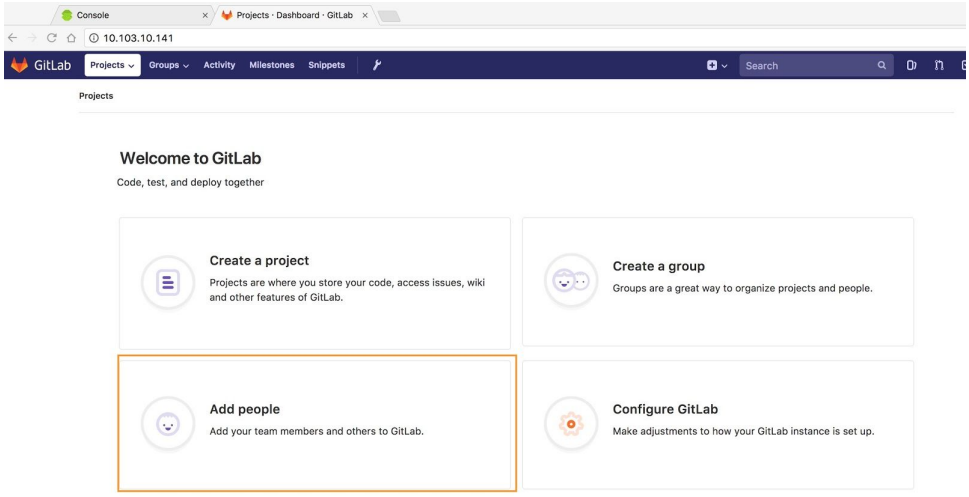


When the password is set, log in with the **root** account in order to configure GitLab.

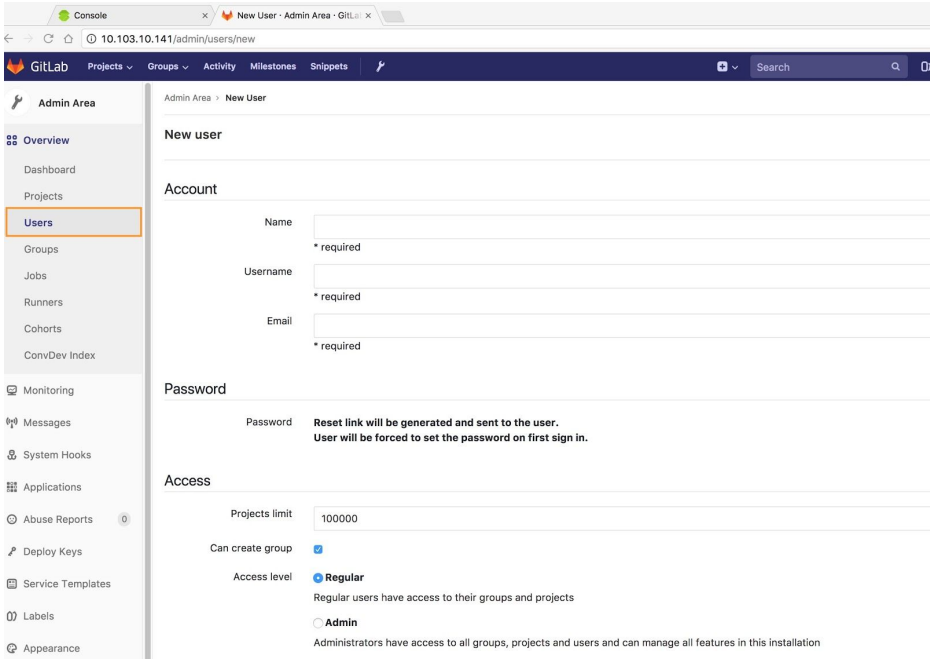
NOTE: For security secondary admin accounts should be created. The root account should not be used for day to day administrative tasks.



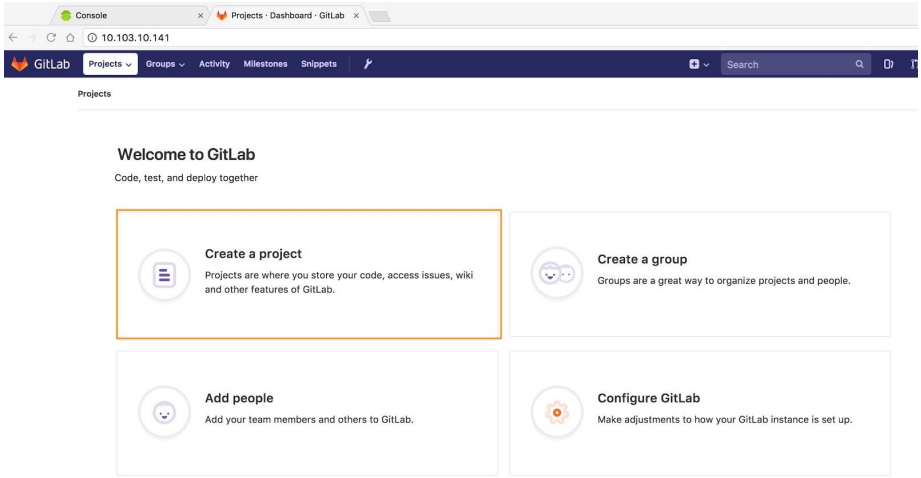
Once logged in click on the **Add People** panel, and enter in the gitlab users accounts. Also, make sure to add at least one secondary admin account.



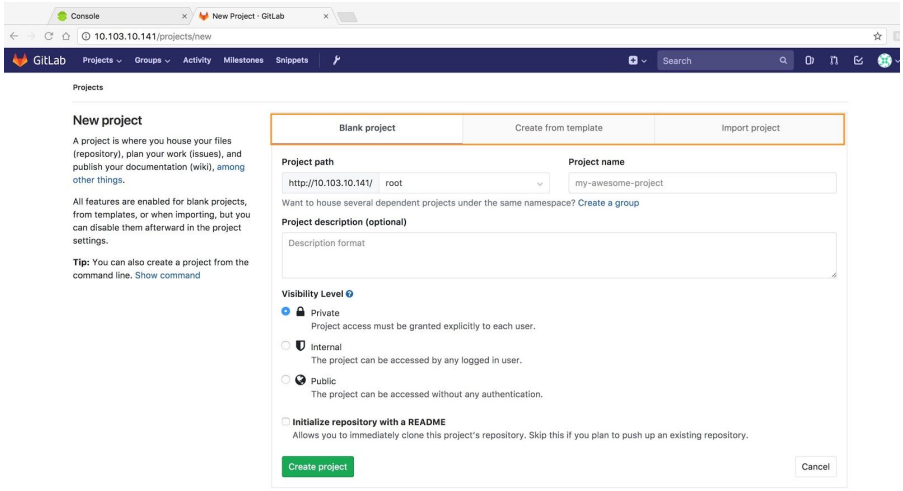
Once in the user panel, new users can be added.



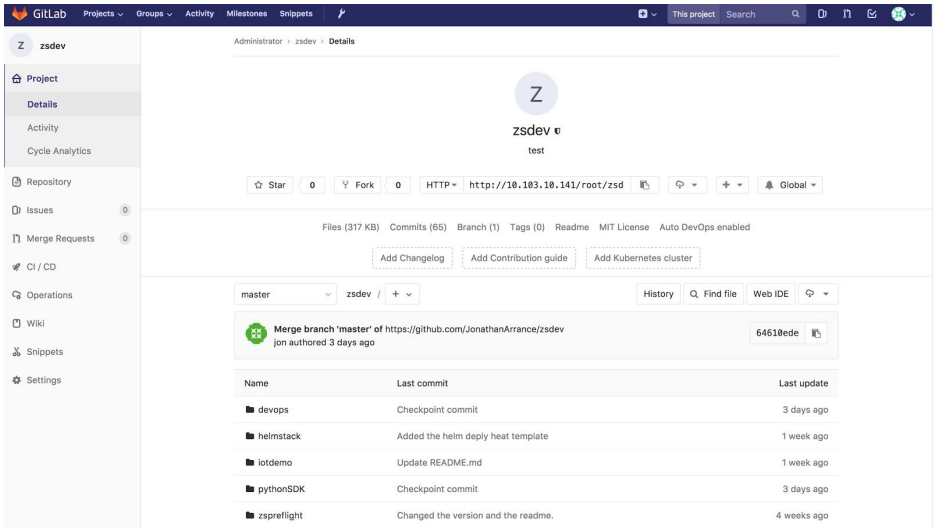
Once the user accounts are added, code repositories can be created or imported.



Once in the **Create a project** interface, select the proper create option, and build the project.



When the repository is created code can be pushed, pulled and merged into the repository.



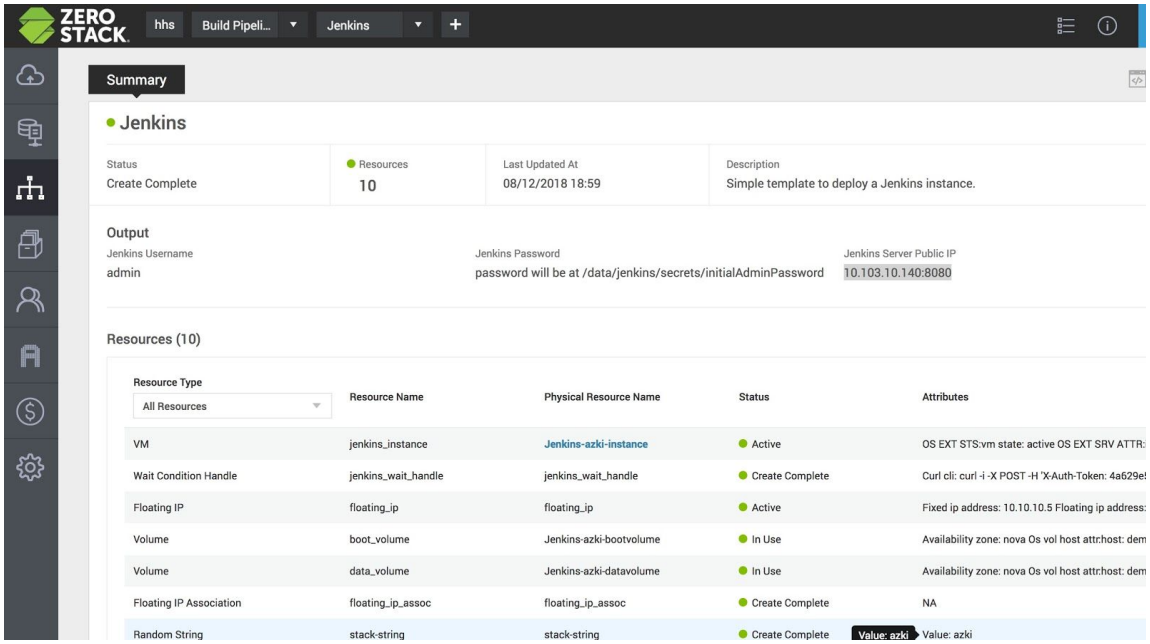
Jenkins

Why use Jenkins

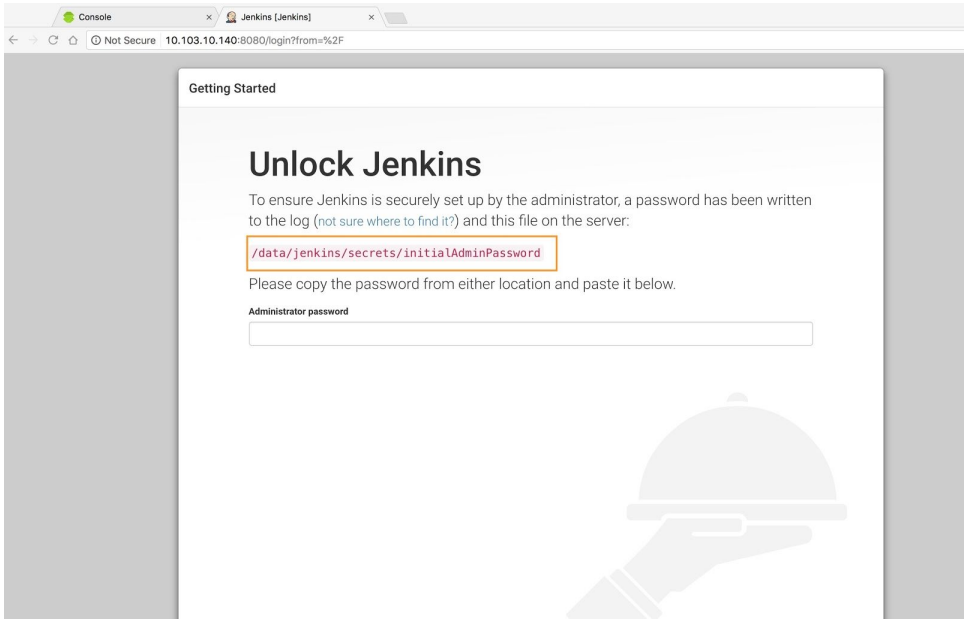
We are going to use Jenkins in this deployment since it is the best know CI automation environment available. There are numerous plugins, and active community, and a commercial version available. Jenkins also has a track record an is used in thousands of production environments.

Configure Jenkins

In order to login to the Jenkins interface, use the URL located in the Jenkins app description.



When Jenkins first comes online, the host VM will need to be logged into in order to get the initial admin access password. Each deployment will have a different default admin password. This password can be changed at any time.

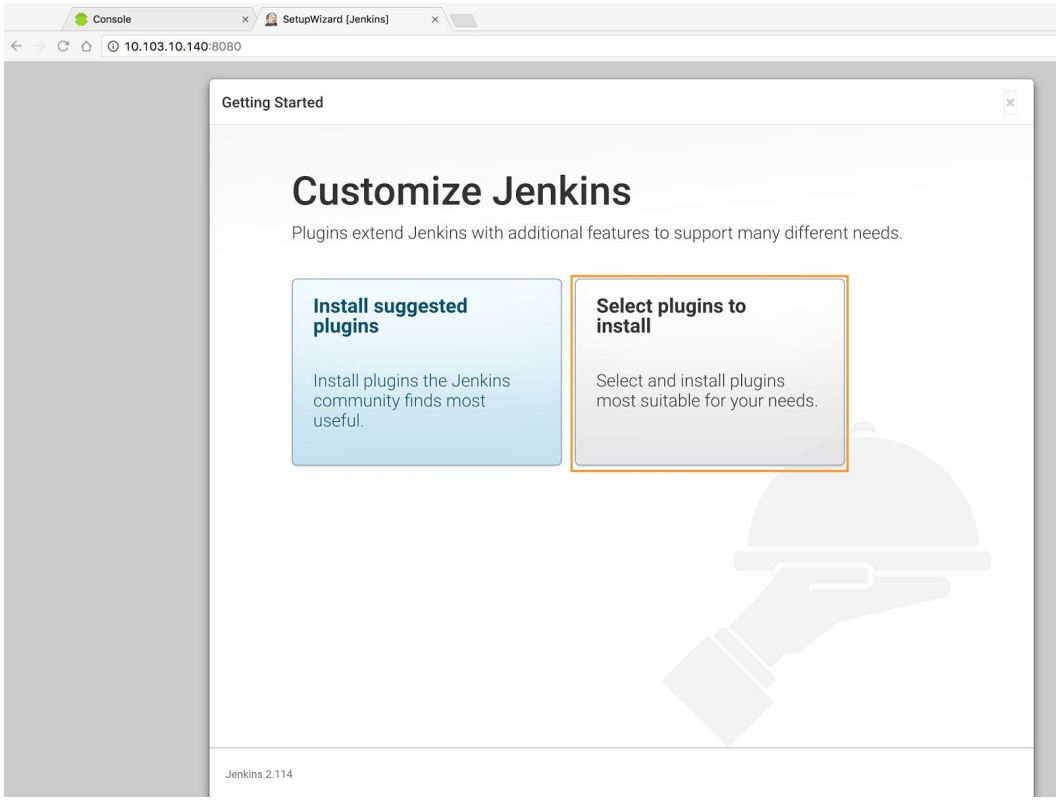


NOTE: All of the VMs use the same ssh key built in the control project.

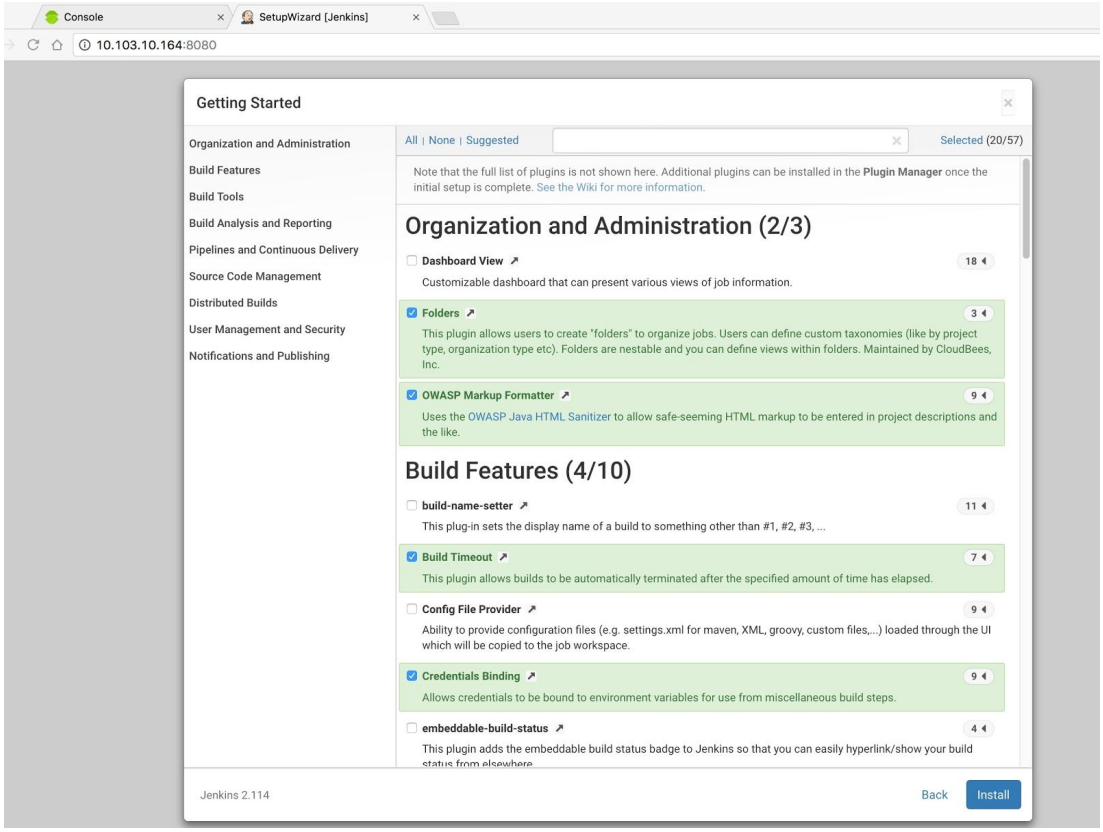
```
# chmod 0400 "keypairname"
# ssh -i "keypairname" username@ipaddress.com
```

```
Downloads — root@jenkins:/home/centos — ssh -i ./hhs_devops_
~ — builder@zsdev: ~/zsdev/devops/devops_deploy — ssh -l builder 192.168.10.224
zerobook:Downloads jonathan$ ssh -i ./hhs_devops_keypair.txt centos@10.103.10.140
Last login: Sun Aug 12 19:56:55 2018 from 10.30.10.12
[centos@jenkins ~]$ sudo su
[root@jenkins centos]# more /data/jenkins/secrets/initialAdminPassword
589177bdd6604b948e2cc3c5f6a0d08d
[root@jenkins centos]#
```

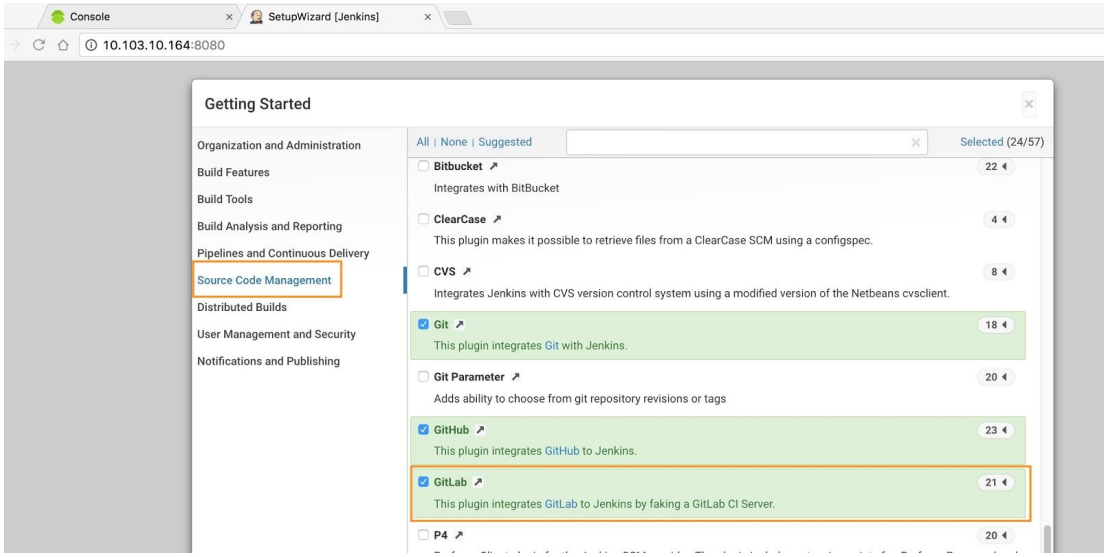
Once the code is entered, Jenkins will be unlocked. To configure Jenkins select the plugins to install.



Since all environments are different the plugins needed for an environment may be different. The plugins selected are common to most environments. Once all of the plugins needed are selected, click on the install button.

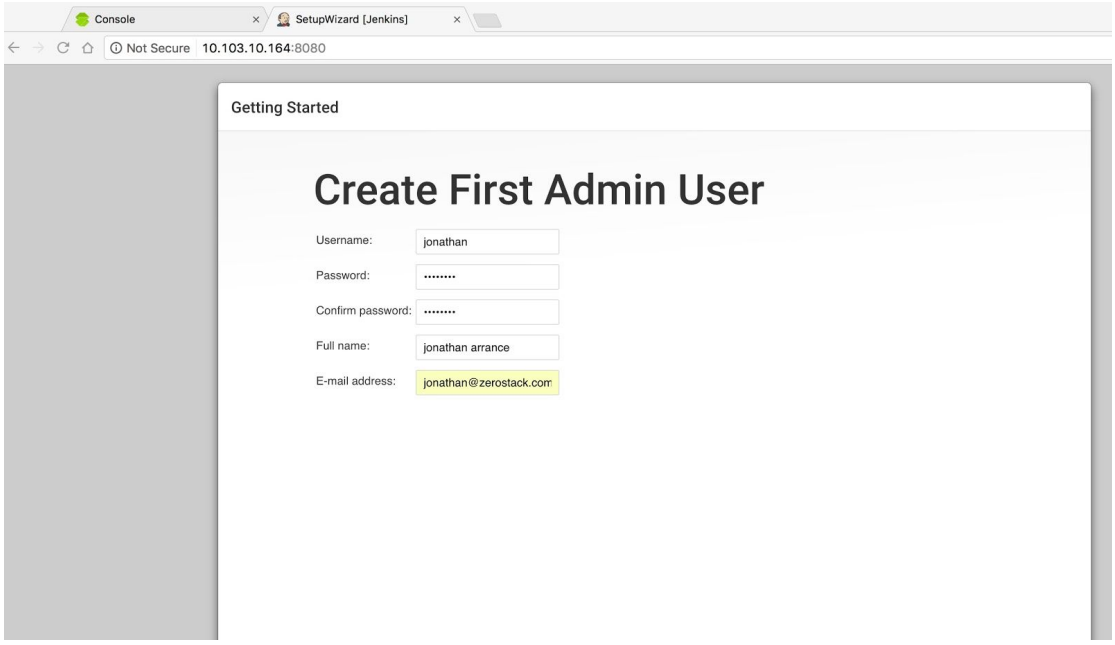


In this example make sure that GitLab is selected as one of the available code repositories.



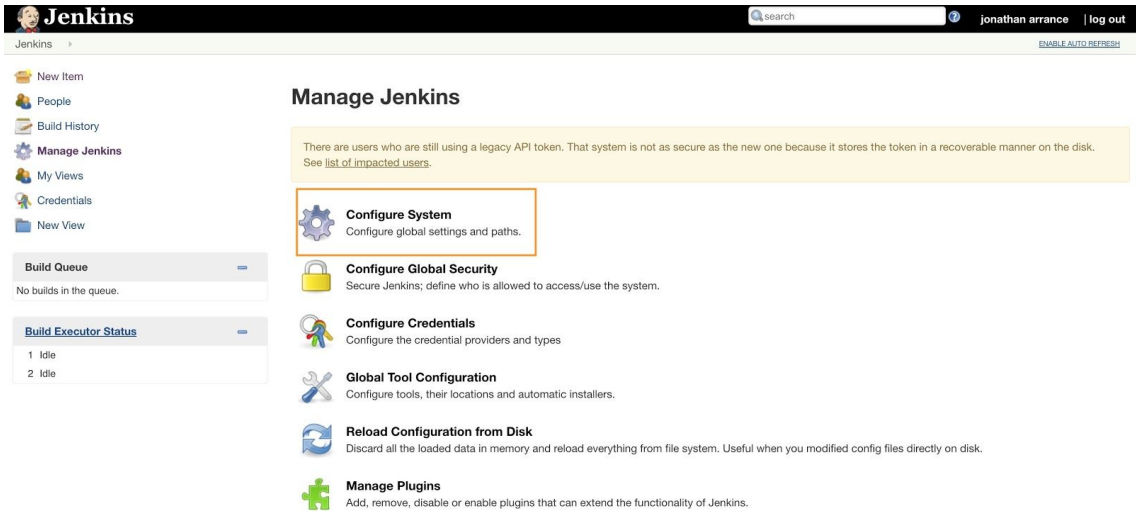
Once all of the plugins have been successfully installed, the create first admin user page will appear.

NOTE: If the plugins fail it may be because jenkins needs to be updated. Finish the setup and then follow the Jenkins update procedure.



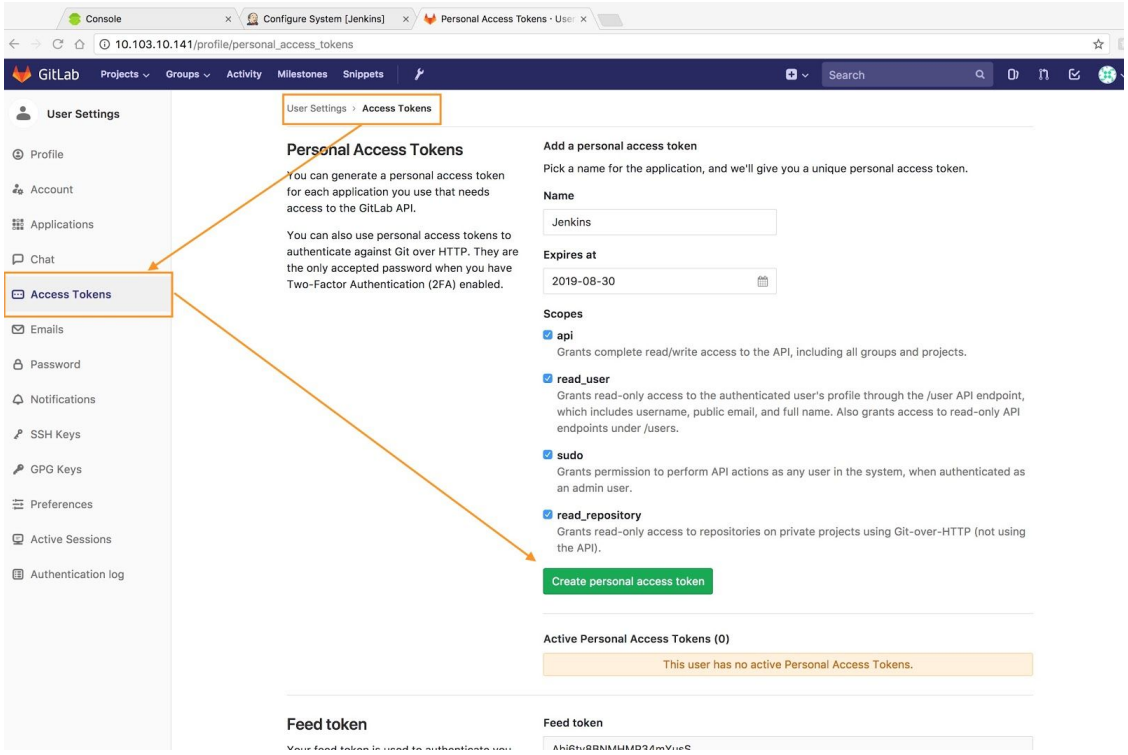
Once Jenkins is set up and ready, the system can be configured and new jobs can be created.

The first task that should be completed is connecting Jenkins to a code repo.

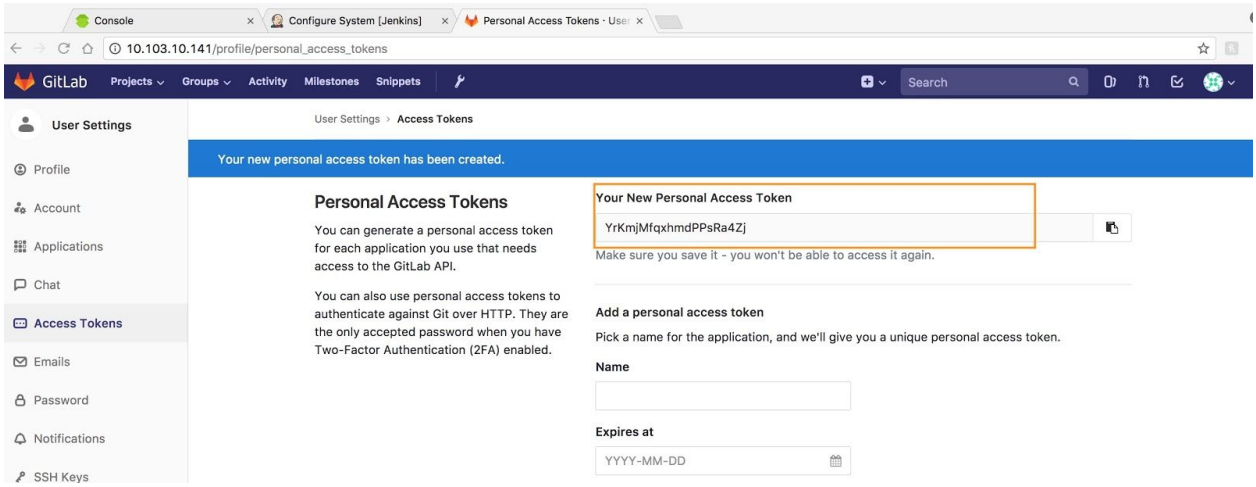


In order to connect Jenkins to the GitLab code repository, the Jenkins system will need an API token from GitLab.

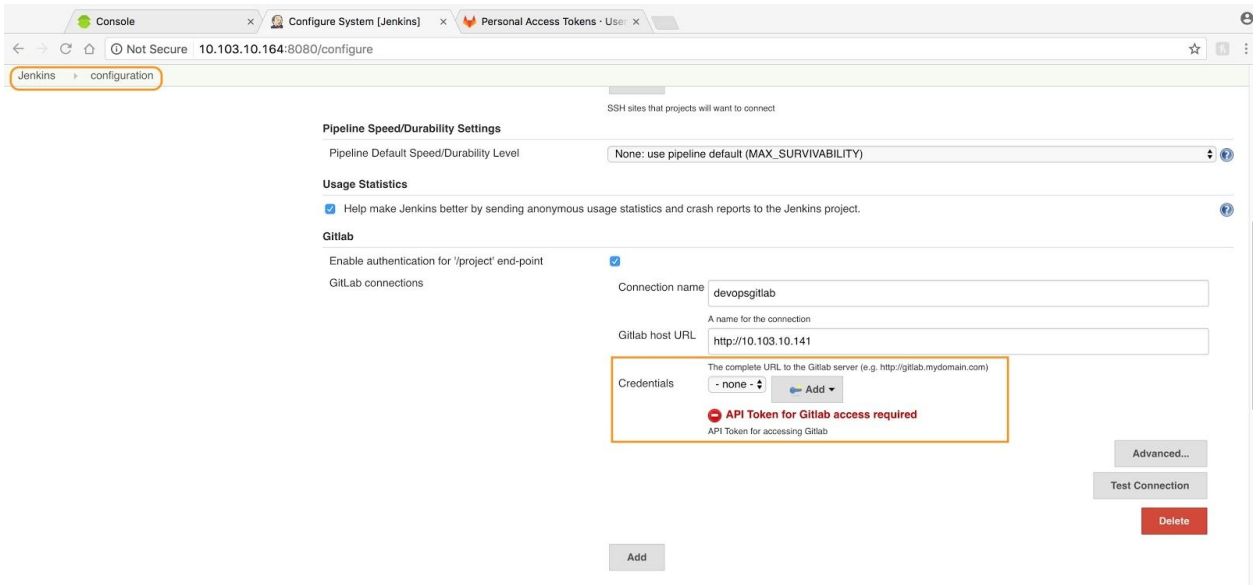
Once logged into the GitLab interface, goto the user settings to generate the access token.



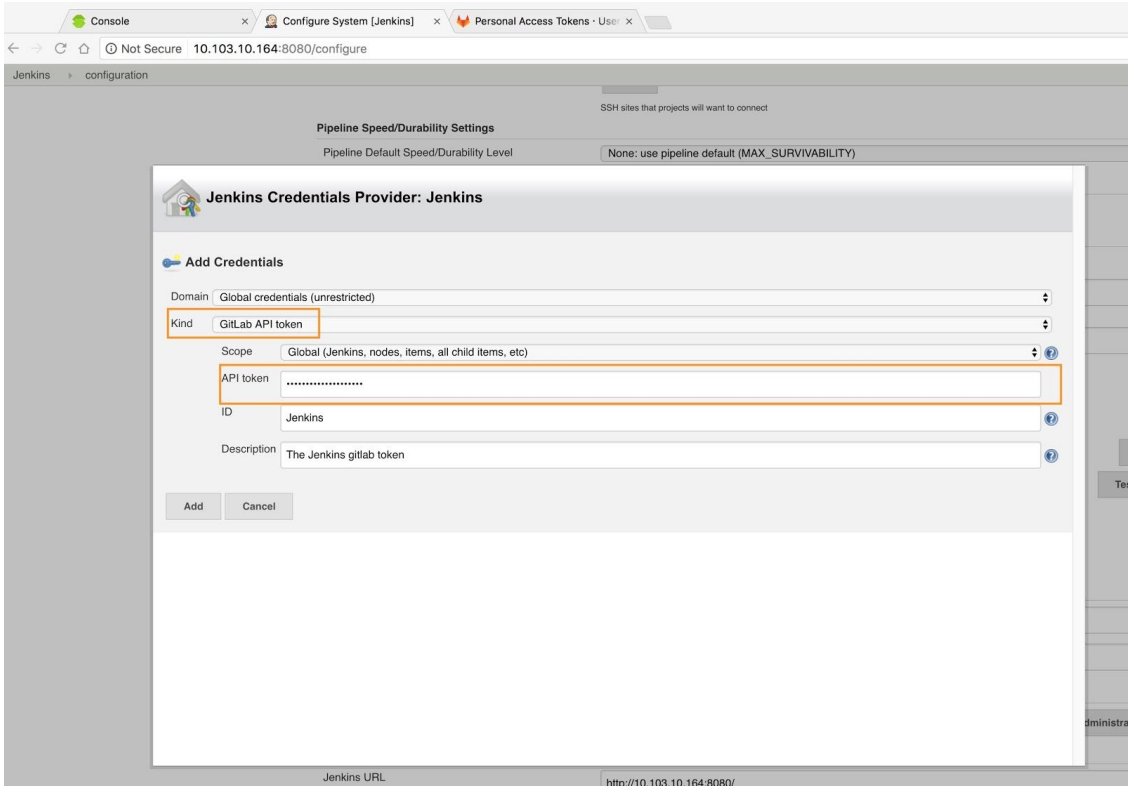
When the **create personal access token** button is clicked the token will be presented. Copy the token so it can be pasted into the Jenkins configuration.



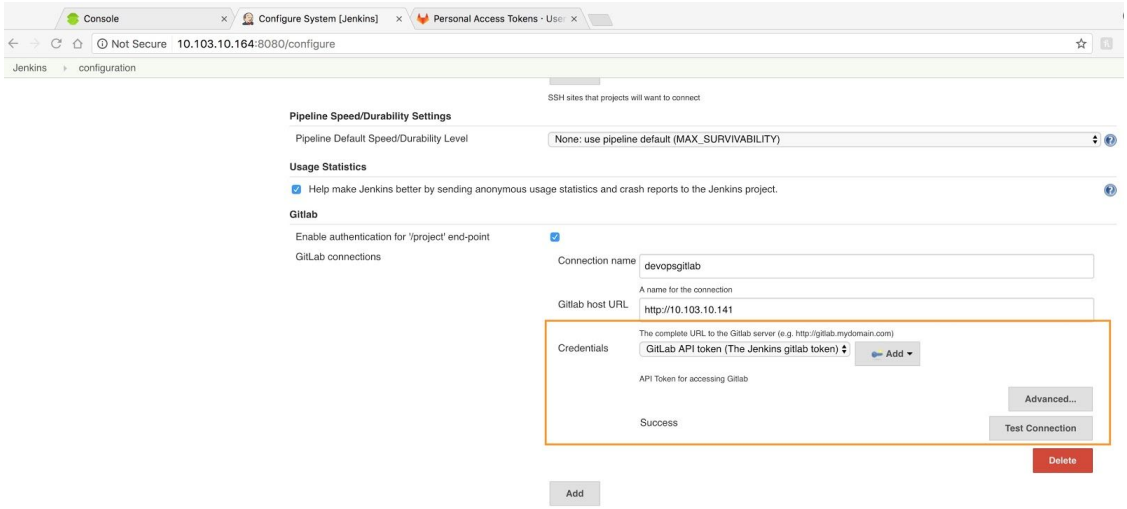
Flip back over to the Jenkins interface, and configure the GitLab code repo. Go into the Jenkins configuration, scroll to the GitLab section. Fill out the connection name, GitLab URL, and click the **Add** button to load the token from GitLab.



When the interface to configure the token appears make sure you paste the GitLab token into the API token field.



Once the token is successfully loaded, the page will reload with the GitLab token set. If the token is set properly the Test Connection will return a success.



Now that GitLab and Jenkins are connected, a new Jenkins code build pipeline can be created. Building the pipeline for your project is outside of the scope of this document. Please refer to the appendix.

Ansible

What is Ansible

Ansible is an automation engine that can be used to automate tasks in physical and virtual environments. Ansible is very user friendly and does not require any knowledge of programming in order to use it. Ansible is a two part solution, The Playbooks, which contains the commands for Ansible to execute on the target system, and the Ansible Engine, which runs the playbooks. The Ansible Engine is a Python application and is now under the RedHat umbrella.

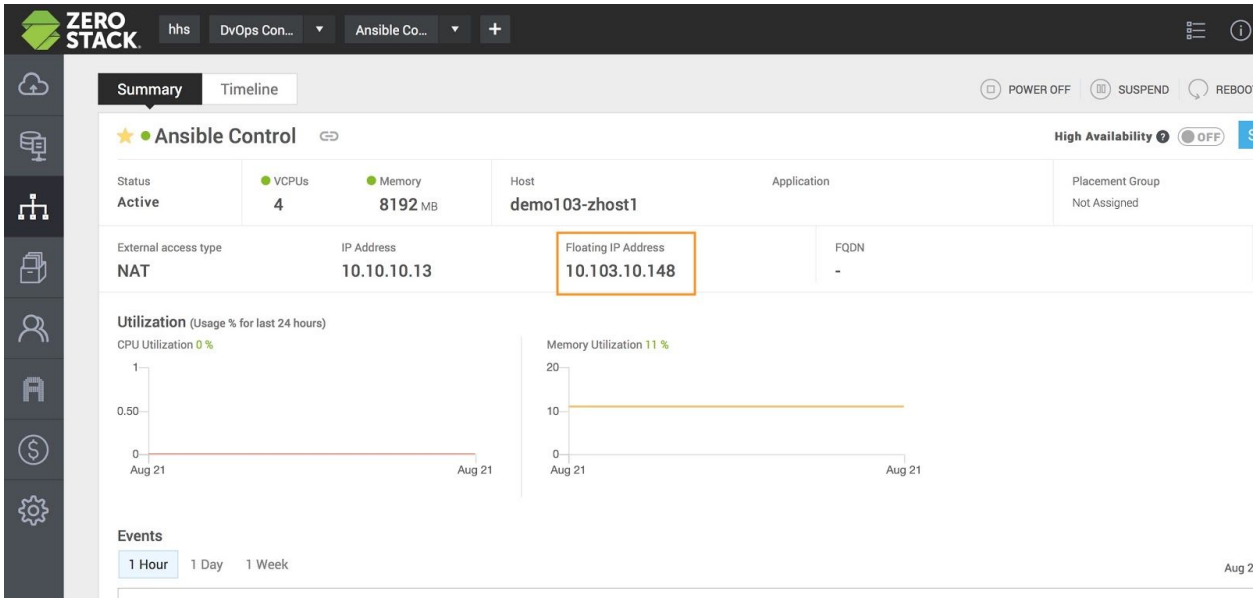
Why use Ansible with Devops

Ansible is ideal in a devops environment because it allows an administrator to automate tasks from end to end. The Ansible platform is agentless, which means no code runs on the target system, and is very easy to maintain. Ansible also has a very active community around it and has an increasing amount of mindshare in the DevOps universe.

Upload Playbooks to Ansible VM

In order to use Ansible you will need to create playbooks. The playbooks will tell the Ansible Engine what commands to run in order to configure or manage the target system.

In order to log into the Ansible VM, download the default ssh key created with the control project.



Once the floating IP is obtained, sftp can be used to upload the playbooks.

```
# sftp -i default_key username@10.103.10.148
sftp> mkdir playbooks
sftp> put my_playbook
sftp> exit
```

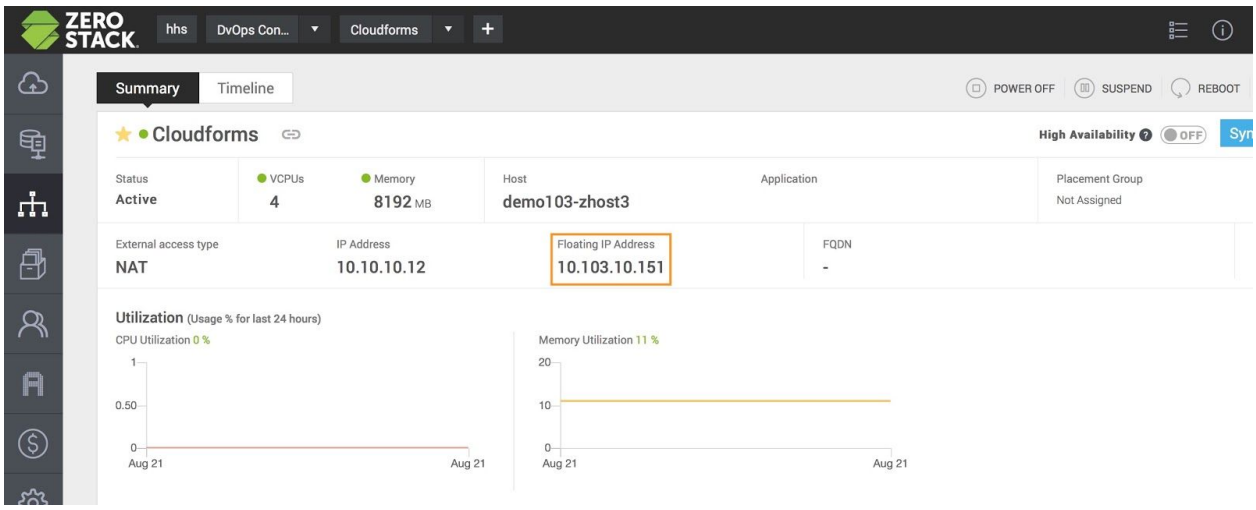
For tips on building out playbooks and configuring the Ansible Engine refer to the appendix.

CloudForms/ManageIQ

The deploy.py script deploys a Cloud Management Platform(CMP) that can be used to administer the DevOps environment from a single interface. ManageIQ is the upstream release of Cloudforms and it integrates well with the Zerostack API, Ansible, and OpenShift.

ManageIQ Interface

To get the CMP URL get the public facing IP of the CloudForms VM.



The Login URL will be

https://10.103.10.151:8443

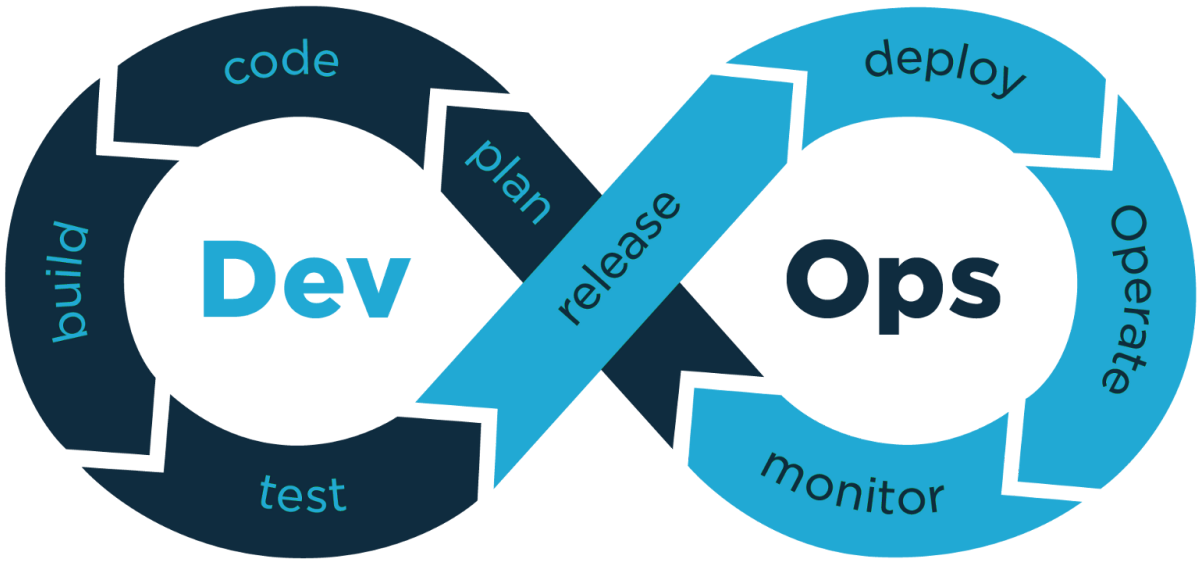
Default login
Username: admin
Password: smartvm

Once logged in the components can be added.

Appendix

Example DevOps flow

The following diagram show the continuous workflow inherent in a devops environment. The diagram is a representation of the continuous iteration, continuous delivery methodology DevOps seeks to achieve.



Source: https://medium.com/tech-tajawal/devops-in-a-scaling-environment-9d5416ecb928

Ansible

Ansible Install

In order to configure an ansible control VM run the following script either in the created VM, or build a new VM and paste it into the bootable script section.

```
#!/bin/bash

if [ -f '/etc/redhat-release' ]; then
      yum install -y epel-release
      yum install -y git
      yum install -y python-setuptools python-setuptools-devel
      easy_install pip

      pip install --upgrade ansible 2>&1
else
      apt-get update -y
      apt-get install software-properties-common -y
      apt-add-repository ppa:ansible/ansible -y
      apt-get update -y
      apt-get install ansible -y
fi
```

Configure Ansible

Please refer to this section of the Ansible docs to add configurations to Ansible

https://docs.ansible.com/ansible/2.5/installation_guide/intro_configuration.html

Ansible Playbooks

Ansible playbooks are the commands that the Ansible Engine runs against the target systems.

https://docs.ansible.com/ansible/latest/user_guide/playbooks.html

CloudForms

CloudForms/ManageIQ install

CloudForms is a cross platform cloud management platform that will enable an end user to manage their cloud environment across local and remote compute resources. This script will install ManageIQ, which is the upstream of CloudForms, in a docker container.

```
#!/bin/bash

if [ -f '/etc/redhat-release' ]; then
      yum install -y epel-release
      yum install -y git
      yum install -y easy_install
      easy_install pip
      yum install -y yum-utils device-mapper-persistent-data lvm2
      yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
      yum install -y docker-ce
      systemctl start docker
      systemctl enable docker
      service docker start
      docker pull manageiq/manageiq:gaprindashvili-4
      docker run --privileged -d -p 8443:443 manageiq/manageiq:gaprindashvili-4
else
      apt-get update -y
      apt-get install apt-transport-https ca-certificates curl
software-properties-common -y
      curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
      apt-key fingerprint 0EBFCD88
      add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

```
        apt-get update -y
        apt-get install docker-ce -y
        service docker start
        docker pull manageiq/manageiq:gaprindashvili-4
        docker run --privileged -d -p 8443:443 manageiq/manageiq:gaprindashvili-4
fi
```

Jenkins

Update Jenkins

In order to update Jenkins download the updated Jenkins.war file from the Jenkins interface.

Once the war is downloaded use sftp to upload the file to the vm the hosts the Jenkins app.

```
# sftp -i ./sshkey user@url
# cd /usr/share/jenkins
# sftp> put jenkins.war
```

Jenkins Config Docs

Configuring Jenkins is outside of the scope of this document. Each environment will be different and the corresponding configuration will also be different.

https://jenkins.io/doc/