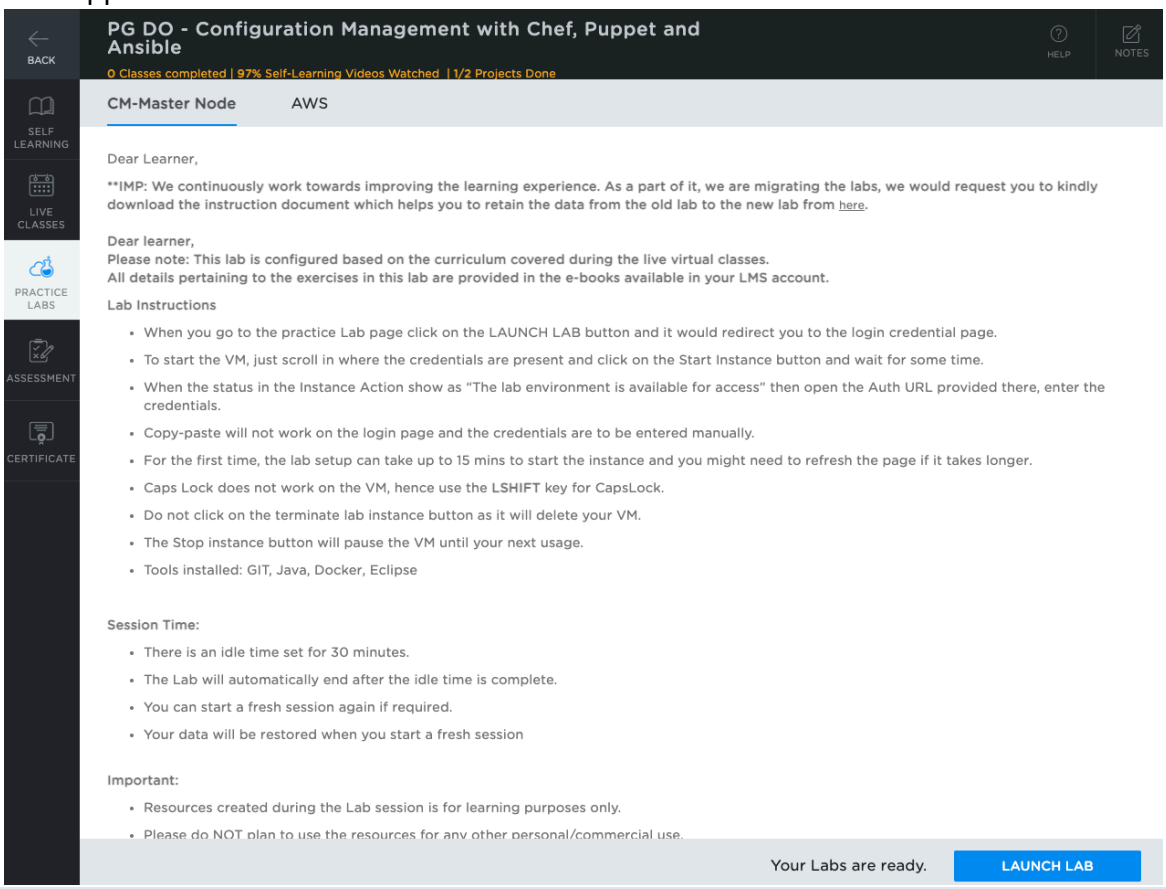
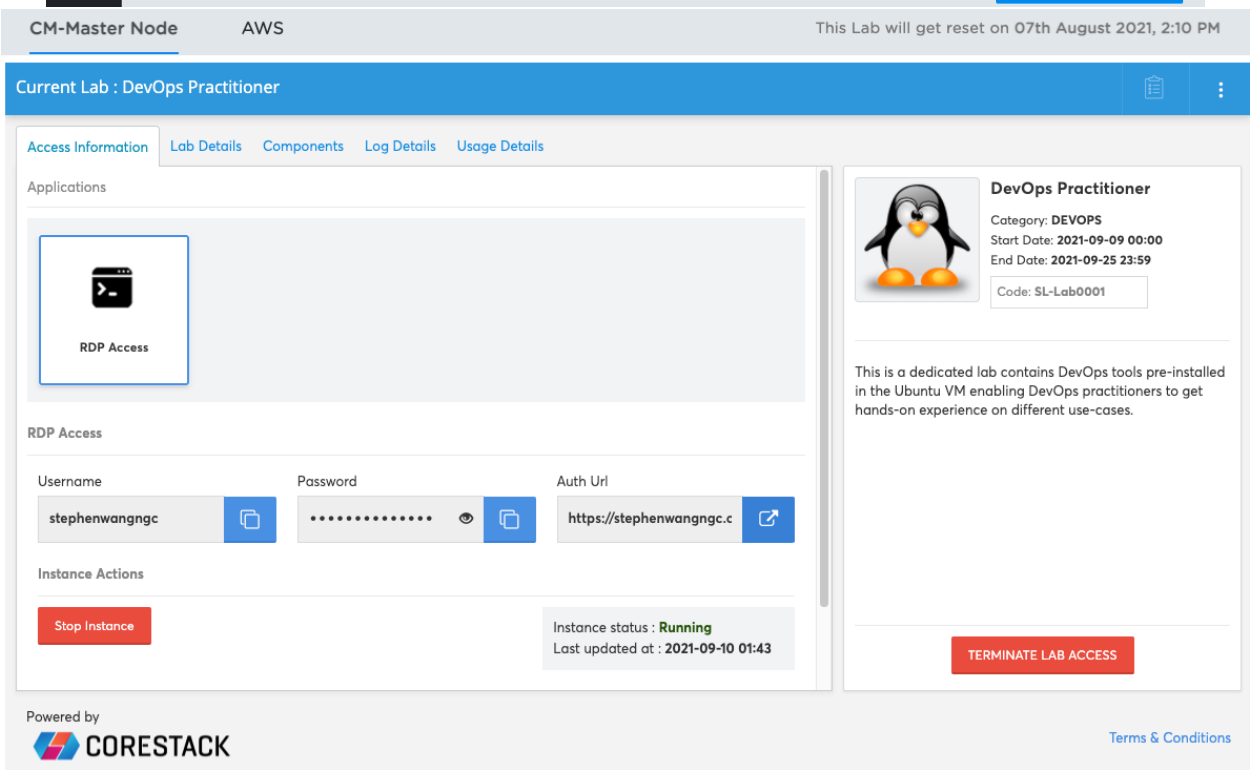
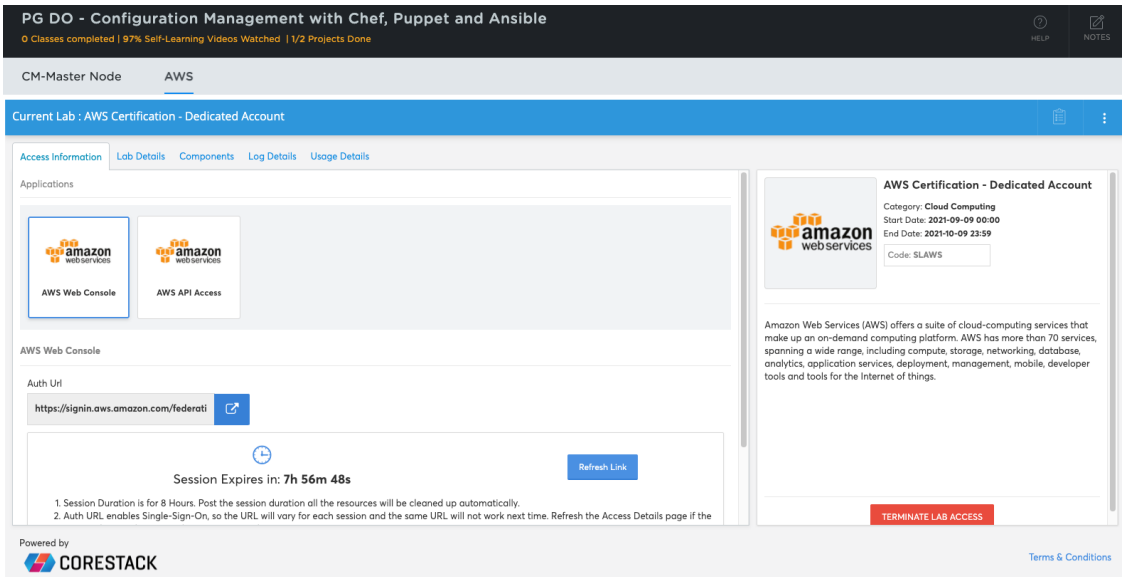
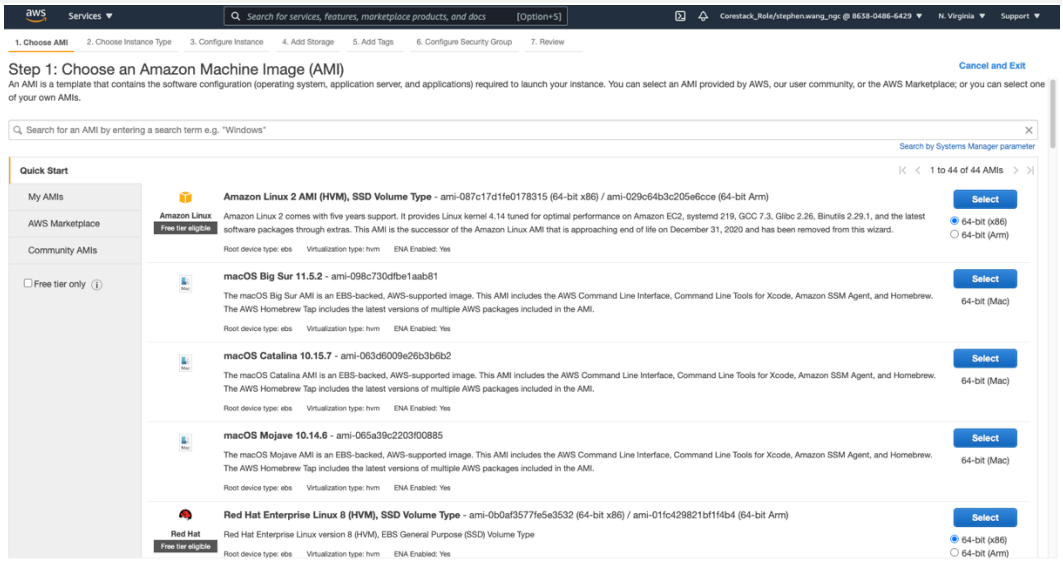
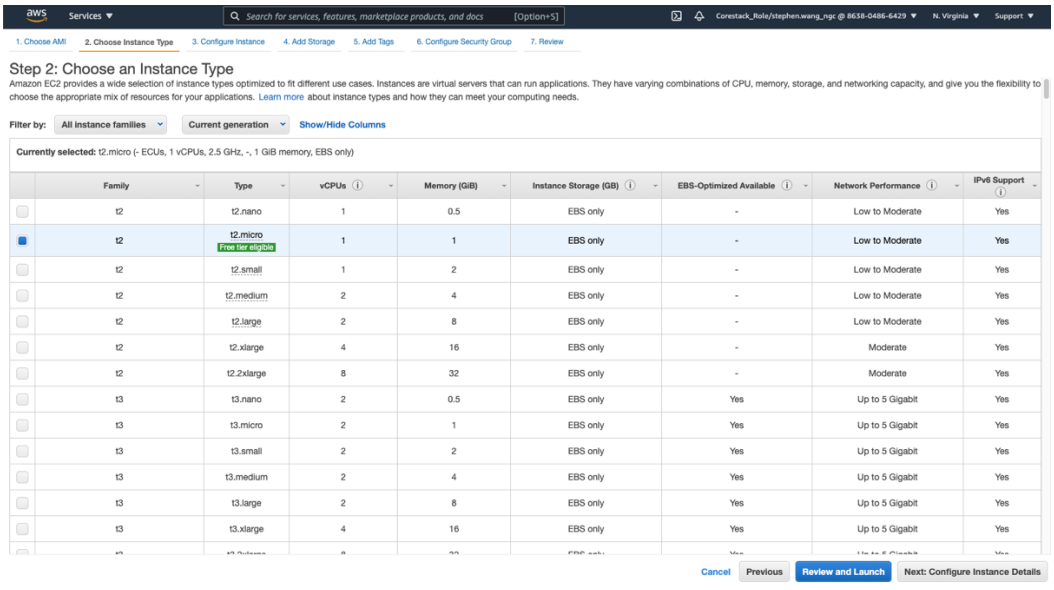
1. Lauch CM-Master Node Lab in Course 4: PG DO – Configuration Management with Chef, Puppet and ansible

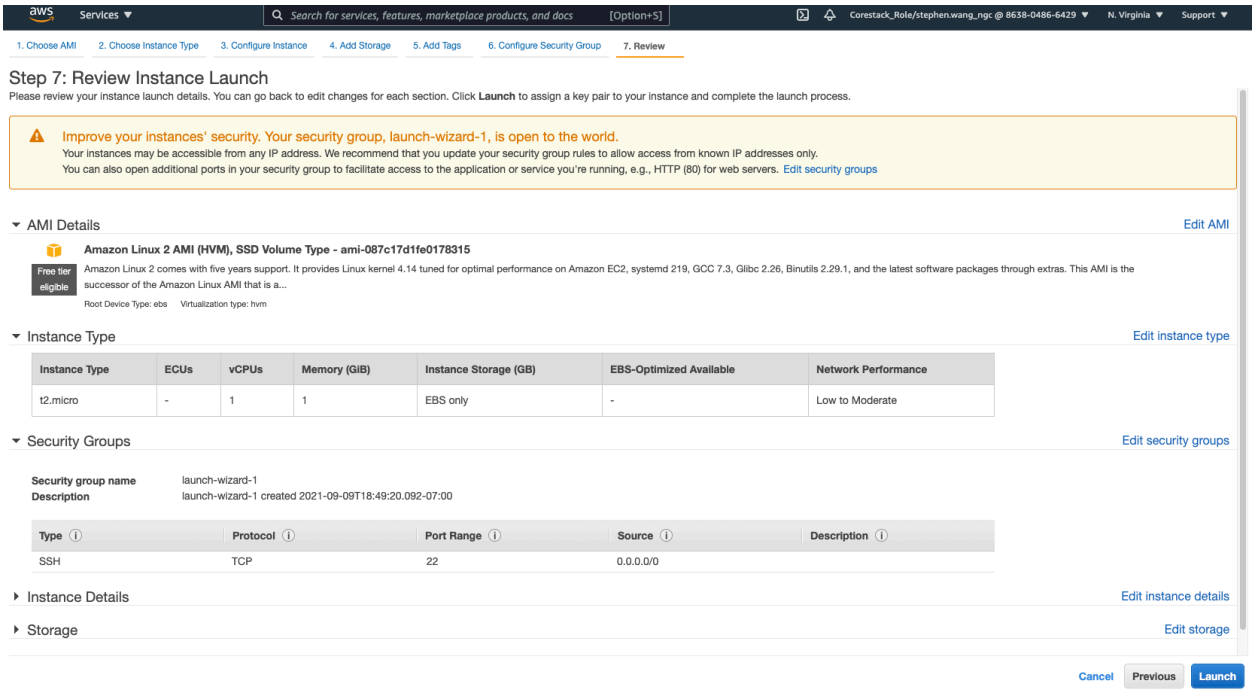





PG DO - Configuration Management with Chef, Puppet and Ansible
0 Classes completed | 97% Self-Learning Videos Watched | 1/2 Projects Done
HELP
NOTES
CM-Master Node
AWS
Current Lab : AWS Certification - Dedicated Account
Access Information
Lab Details
Components
Log Details
Usage Details
Applications
AWS Web Console
AWS API Access
AWS Web Console
Auth Url
https://signin.aws.amazon.com/federati
Session Expires in: 7h 56m 48s
Refresh Link
1. Session Duration is for 8 Hours. Post the session duration all the resources will be cleaned up automatically.
2. Auth URL enables Single-Sign-On, so the URL will vary for each session and the same URL will not work next time. Refresh the Access Details page if the
AWS Certification - Dedicated Account
Category: Cloud Computing
Start Date: 2021-09-09 00:00
End Date: 2021-10-09 23:59
Code: SLAWS
Amazon Web Services (AWS) offers a suite of cloud-computing services that make up an on-demand computing platform. AWS has more than 70 services, spanning a wide range, including compute, storage, networking, database, analytics, application services, deployment, management, mobile, developer tools and tools for the Internet of things.
TERMINATE LAB ACCESS
Powered by
CORESTACK
Terms & Conditions


Services
Search for services, features, marketplace products, and docs [Option+S]
N. Virginia
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
Search for an AMI by entering a search term e.g. "Windows"
Search by Systems Manager parameter
Quick Start
1 to 44 of 44 AMIs
My AMIs
AWS Marketplace
Community AMIs
Free tier only
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-087c17d1fe0178315 (64-bit x86) / ami-029c64b3c205e6cce (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.
Select
64-bit (x86)
64-bit (Arm)
macOS Big Sur 11.5.2 - ami-098c730dfbe1aab81
The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.
64-bit (Mac)
macOS Catalina 10.15.7 - ami-063d6009e26b3b6b2
macOS Mojave 10.14.6 - ami-065a39c2203f00885
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0b0af3577fe5e3532 (64-bit x86) / ami-01fc429821bf1f4b4 (64-bit Arm)
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
Free tier eligible


Services
Search for services, features, marketplace products, and docs [Option+S]
N. Virginia
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 2: Choose an Instance Type
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.
Filter by: All instance families Current generation Show/Hide Columns
Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)
Family | Type | vCPUs | Memory (GiB) | Instance Storage (GB) | EBS-Optimized Available | Network Performance | IPv6 Support
t2 | t2.nano | 1 | 0.5 | EBS only | - | Low to Moderate | Yes
t2 | t2.micro Free tier eligible | 1 | 1 | EBS only | - | Low to Moderate | Yes
t2 | t2.small | 1 | 2 | EBS only | - | Low to Moderate | Yes
t2 | t2.medium | 2 | 4 | EBS only | - | Low to Moderate | Yes
t2 | t2.large | 2 | 8 | EBS only | - | Low to Moderate | Yes
t2 | t2.xlarge | 4 | 16 | EBS only | - | Moderate | Yes
t2 | t2.2xlarge | 8 | 32 | EBS only | - | Moderate | Yes
t3 | t3.nano | 2 | 0.5 | EBS only | Yes | Up to 5 Gigabit | Yes
t3 | t3.micro | 2 | 1 | EBS only | Yes | Up to 5 Gigabit | Yes
t3 | t3.small | 2 | 2 | EBS only | Yes | Up to 5 Gigabit | Yes
t3 | t3.medium | 2 | 4 | EBS only | Yes | Up to 5 Gigabit | Yes
t3 | t3.large | 2 | 8 | EBS only | Yes | Up to 5 Gigabit | Yes
t3 | t3.xlarge | 4 | 16 | EBS only | Yes | Up to 5 Gigabit | Yes
Cancel
Previous
Review and Launch
Next: Configure Instance Details

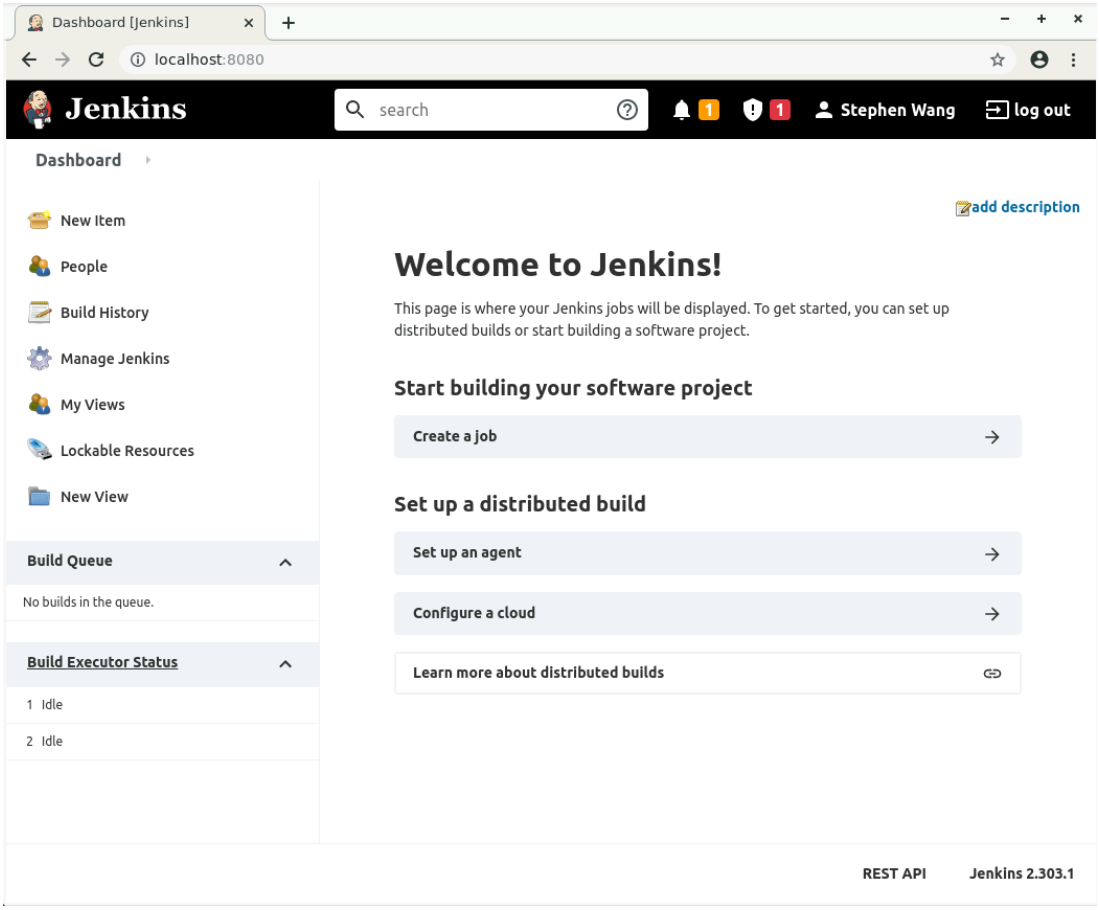


2. Use Github to create and clone capstone project to CM-Master Note

```
stephenwangngc@stephenwangngc:~$ ls
Desktop  thinclient_drives
stephenwangngc@stephenwangngc:~$ git clone https://github.com/stephengineer/DevO
psCapstone.git
Cloning into 'DevOpsCapstone'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
stephenwangngc@stephenwangngc:~$
```

3. Set up the Jenkins server in master or slave architecture

```
stephenwangngc@stephenwangngc:~/DevOpsCapstone$ sudo systemctl start jenkins
stephenwangngc@stephenwangngc:~/DevOpsCapstone$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Thu 2021-09-09 20:41:48 UTC; 1min 48s ago
     Docs: man:systemd-sysv-generator(8)

Sep 09 20:41:46 stephenwangngc systemd[1]: Starting LSB: Start Jenkins at boot time...
Sep 09 20:41:46 stephenwangngc jenkins[31966]: Correct java version found
Sep 09 20:41:46 stephenwangngc jenkins[31966]:  * Starting Jenkins Automation Server jenkins
Sep 09 20:41:46 stephenwangngc su[32022]: Successful su for jenkins by root
Sep 09 20:41:46 stephenwangngc su[32022]: + ??? root:jenkins
Sep 09 20:41:46 stephenwangngc su[32022]: pam_unix(su:session): session opened for user jenkins by (uid=0)
Sep 09 20:41:48 stephenwangngc jenkins[31966]:    ...done.
Sep 09 20:41:48 stephenwangngc systemd[1]: Started LSB: Start Jenkins at boot time.
Sep 09 20:43:06 stephenwangngc systemd[1]: Started LSB: Start Jenkins at boot time.
stephenwangngc@stephenwangngc:~/DevOpsCapstone$
```

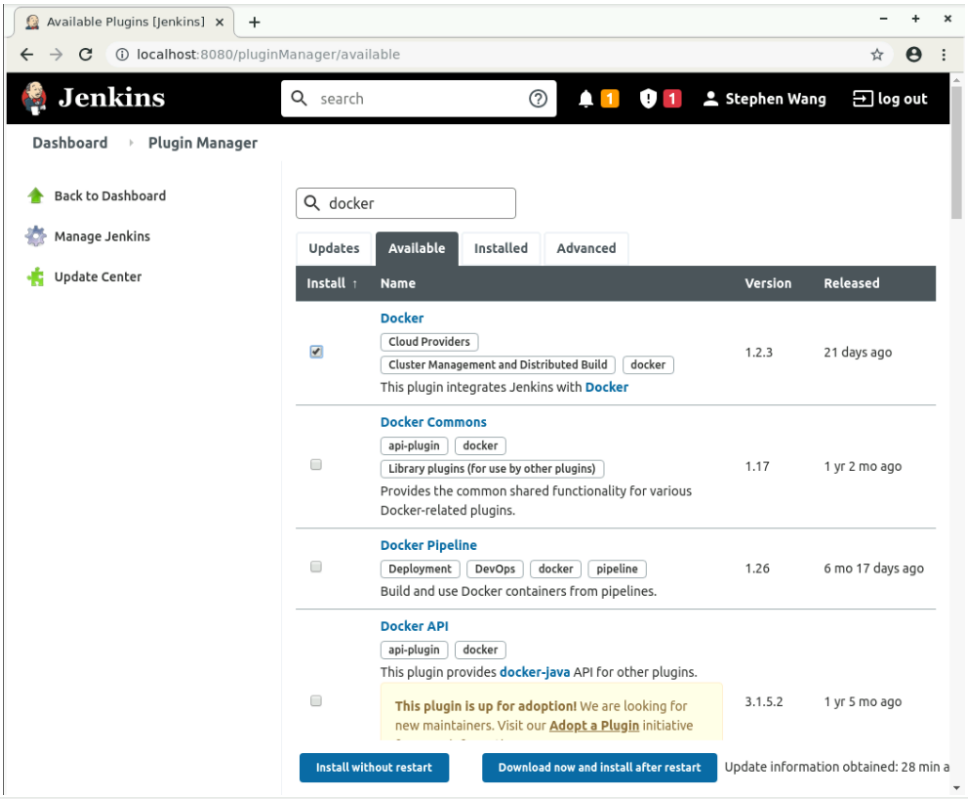


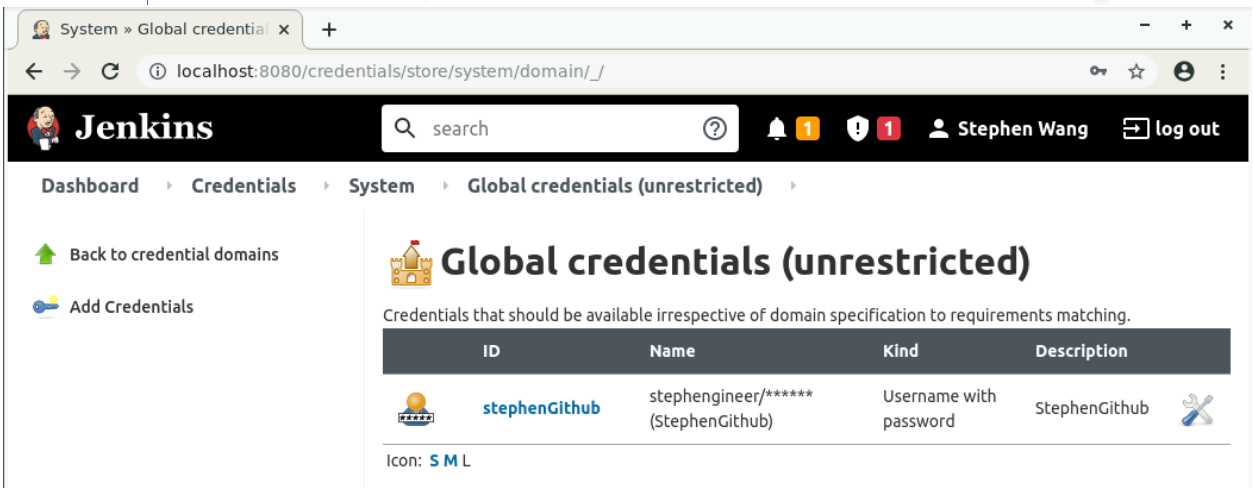
4. Build an image using the artifacts and deploy them on containers
5. Remove the container stack after completing the job

48 lines (46 sloc) | 1.05 KB — Raw | Blame

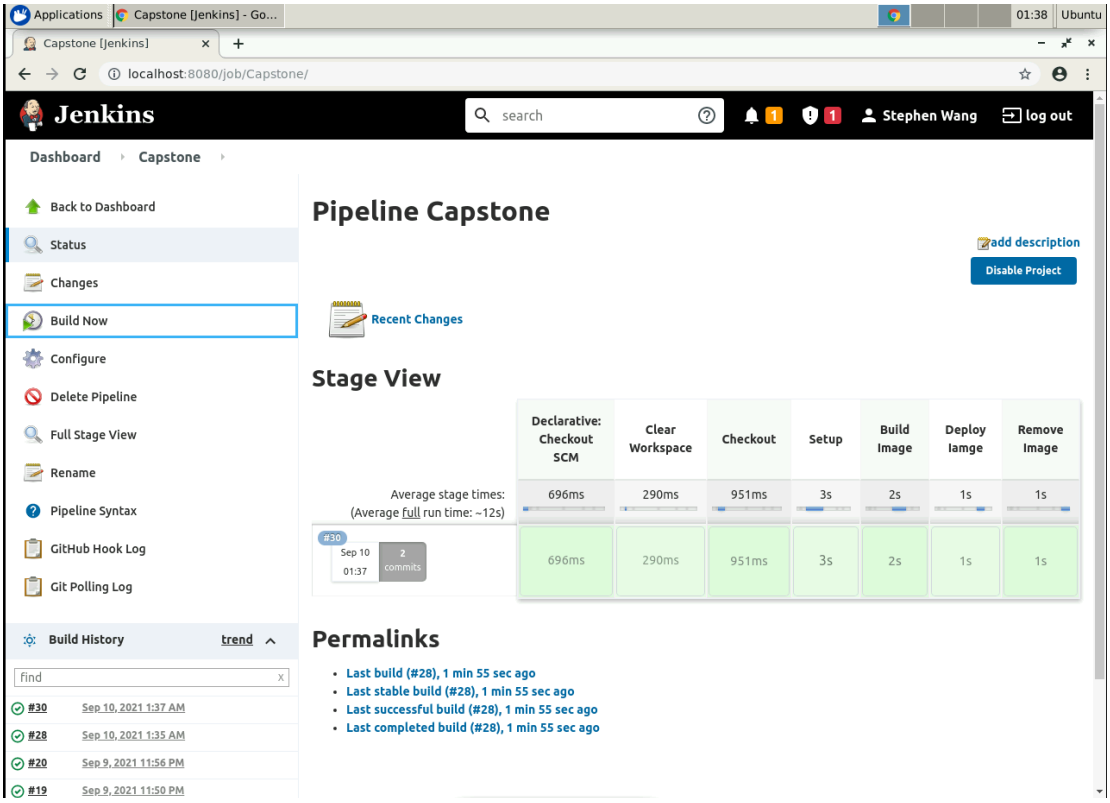
```
pipeline {
    environment {
        registry = "stephenwangmath/devopscapstone"
        registryCredential = 'dockerhub'
        dockerImage = ''
    }

    agent any

    stages {
        stage('Clear Workspace') {
            steps {
                deleteDir()
            }
        }
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage('Setup') {
            steps {
                sh 'npm install'
            }
        }
        stage('Build Image') {
            steps {
                script {
                  dockerImage = docker.build registry + ":$BUILD_NUMBER"
                }
            }
        }
        stage('Deploy Iamge') {
            steps {
                script {
                    docker.withRegistry( '', registryCredential ) {
                        dockerImage.push()
                    }
                }
            }
        }
        stage('Remove Image') {
            steps{
                sh "docker rmi $registry:$BUILD_NUMBER"
            }
        }
    }
}
```

6. Use the Jenkins plugins to perform the computation part on the Docker containers





7. Create Jenkins pipeline script



8. Use the GIT web hook to schedule the job on check-in or poll SCM