

SWE 645 – ASSIGNMENT 2

TEAM MEMBERS: (team of 2)

1. SANJANA GOVINDU | G01380306 | sgovindu@gmu.edu
2. ADITEE GADDAM | G01363937 | agaddam@gmu.edu

URLS for Assignment:

- Github Link - <https://github.com/sanjana-govindu/SWE645/>
- Video Submission:
<https://drive.google.com/file/d/1hjEHWNLjMZAaNpfT83fDcz8S3qm09xmW/view?usp=sharing>
- Containerized Web Application link:
<https://hub.docker.com/repository/docker/sanjanagovindu/swe645-a2/general>

Prerequisites/ Requirements:

1. Github and google account for GCP - google cloud program
2. Tomcat installed on local machine
3. Docker Desktop installed on local machine to build the initial docker image.
4. Account on Docker Hub.
5. AWS running on AWS Academy Learner Lab (link provided by the professor)
6. Jenkins installed on local machine

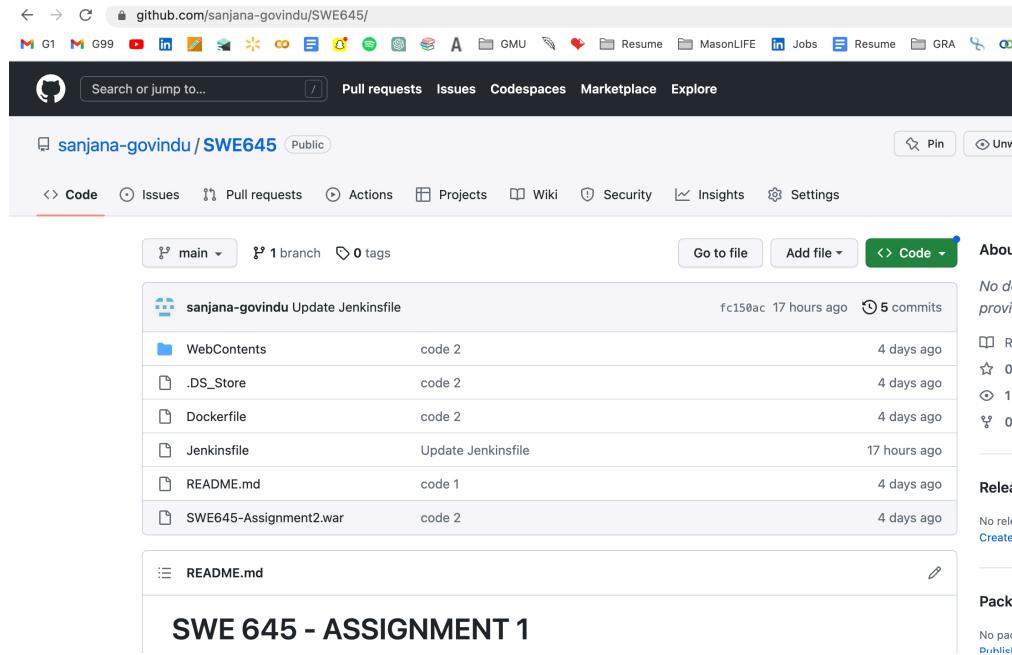
ASSIGNMENT INSTRUCTIONS:

The requirements are as follows:

- Containerize the Web application you developed in Homework 1 – Part 2, using Docker container technology.
- Deploy the containerized application on the open-source container orchestration platform Kubernetes to enable scalability and resiliency of the application. Your baseline configuration includes at least three pods running all the time. You can use Rancher (<https://rancher.com/docs>) to setup Kubernetes cluster. You also have an option to use managed Kubernetes services, such as EKS on Amazon Web Services (AWS) or GKE on Google Cloud Platform (GCP).
- Establish a CI/CD pipeline that includes a Git source code repository at GitHub, and Jenkins for automated build and for the automated deployment of your application on Kubernetes.

PART 1 – PUSH THE CODE ON GITHUB

1. If you already have a github account then create a github repository with with the assignment 1 submission code in WebContents folder and all readme files outside that folder.
2. Use Github Desktop application to commit and push the changes from your local machine into github.
3. This Github link will be useful later for Jenkins part.



PART 2 - CONTAINERIZE WEB APPLICATION

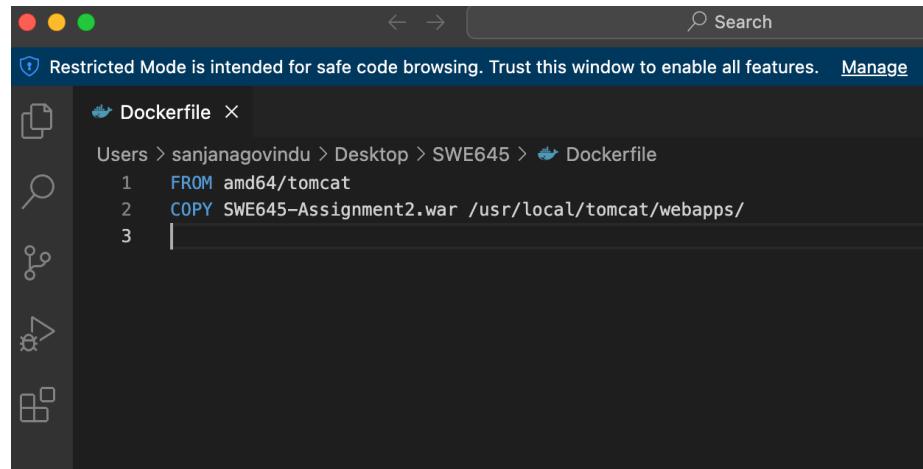
1. To containerize the web application using docker, we need to have docker desktop installed in our local machine i.e., Mac using this link - <https://www.docker.com/products/docker-desktop/>
2. Also, we need to create an account in Docker Hub - <https://hub.docker.com/> with a username (sanjanagovindu) and password of our choice to make our image globally accessible.
3. Create a folder 645 HW 2 in eclipse by creating a dynamic web project and put that WAR file in that folder. To create a war file of the code we can use the below command:
'jar -cvf SWE645-Assignment2.war -C WebContent/.'
4. We used Eclipse IDE to create a file without extension called 'DockerFile' as docker requires the file name to be called in that specific way. The DockerFile should be in the

same folder of that WAR file. It installs tomcat in the container and copies the generated war file into the /usr/local/tomcat/webapps directory.

5. Paste the below in DockerFile:

```
FROM amd64/tomcat
COPY SWE645-Assignment2.war /usr/local/tomcat/webapps/
```

6. In that DockerFile, we used the FROM command to get the initial image required for the docker build.



PART 3 – CREATE IMAGE LOCALLY AND PUSH IT TO DOCKER HUB

1. To access our image globally we need to push it to Docker hub.
2. So, for this we need to create an account in Docker Hub - <https://hub.docker.com/> with a username (sanjanagovindu) and password of our choice.
3. Then, in the locally installed docker desktop we need to login into our account that has been created directly or from command line using the command – ‘**docker login -u <DockerHub_Username>**’.
4. On the command line, use the command: ‘**docker build --tag swe645-a2 .**’ to build a docker image with that name in our local (We can use the name of our own choice).
5. Then we can verify if the image is built properly by running the command “**docker run -itd -p 8182:8080 --name swe645-assignment2 swe645-assignment2**”
6. Then run if the containerized application is working by opening <http://localhost:8182/SWE645-Assignment2/index.html> in your browser like the image below:

The screenshot shows a web page titled "DEPARTMENT OF COMPUTER SCIENCE" with the GMU logo. The main heading is "GMU CS DEPARTMENT - FEEDBACK". Below it is a form with the following fields:

- USERNAME*: [Text input]
- STREET ADDRESS*: [Text input]
- CITY*: [Text input]
- STATE*: [Text input]
- ZIP CODE*: [Text input]
- TELEPHONE NUMBER*: [Text input] (format: ###-###-###)
- E-MAIL*: [Text input] (johnsmith@gmail.com)
- URL*: [Text input] (https://www.github.com/john-smith/)
- DATE OF SURVEY*: [Text input] (MM-DD-YYYY)

7. Then tag the image using the command – “**docker tag swe645-a2 sanjanagovindu/swe645-a2**” and push it to docker hub using the command – “**docker push sanjanagovindu/swe645-a2**”
8. Verify that the image is on Docker Hub as we know that the image is accessible from the internet by logging into Docker hub with our credentials.

The screenshot shows the Docker Hub repository page for "sanjanagovindu/swe645-a2". The repository has one tag, "latest", which was pushed 3 hours ago. There is no description provided. The "Automated Builds" section is available with Pro, Team and Business subscriptions.

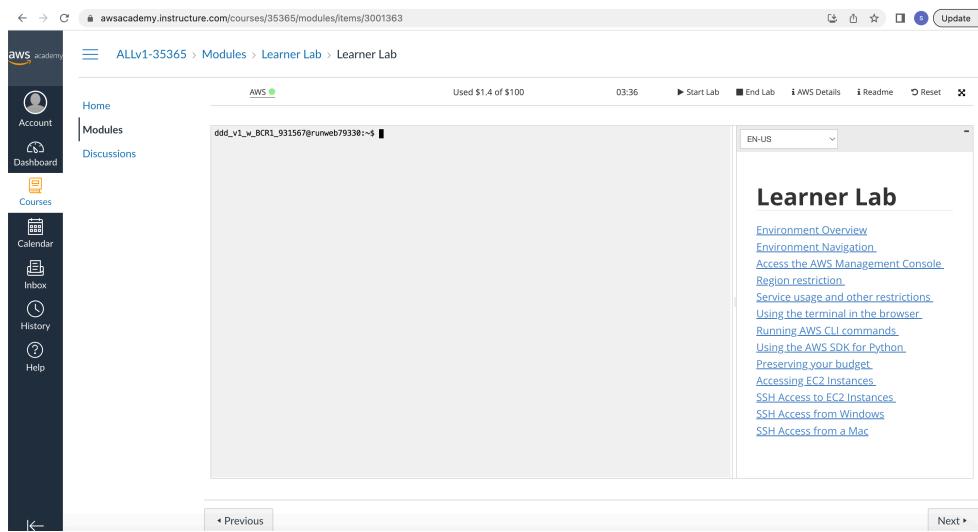
Tag	OS	Type	Pulled	Pushed
latest		Image	2 hours ago	3 hours ago

PART 4 – SETTING UP RANCHER & CREATING A CLUSTER USING RANCHER

Setup Rancher:

1. For this, we need to log on to AWS console <https://aws.amazon.com/> using our credentials or create an account on AWS. In this case we need to use AWS Learner lab for our requirement.

<https://awsacademy.instructure.com/courses/35365?invitation=JMhpNmcZxRQt4I1LnoQuolHzB8cdmYgRtpvwIGJA>



2. Click on Start Lab which will take some time and click on AWS after it is up and navigate to EC2.
3. We need to run that rancher on an Ubuntu EC2 instance with that docker running.
4. To launch that EC2 instance in the AWS Console, click on Services > EC2 and click Instance and ‘Launch instance’.

5. Launch the ubuntu instance with a unique name and t2.large as instance type.

6. Save this key value pair or we can also use the pem file that we have used for our previous assignment. Then allow HTTPS and HTTP traffic from the internet and configure the storage upto 50 for root volume.
7. Now we need to Launch the instance.
8. After the instance is running, we need to get the public IPv4 DNS. Then, open putty by login in with the hostname of that DNS address and authenticating with the PPK key. ssh

-i <YOUR KEY LOCATION> ubuntu@<YOUR PUBLIC IP>. (We might have to change the permission on your key to be read-only.)

PUBLIC IPV4 DNS - ec2-52-70-84-129.compute-1.amazonaws.com

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Tags, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, and Images. The main pane displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
jenkins	i-0a6221165b86f893c	Running	t2.medium	2/2 checks passed	No alarms	us-east-1d	ec2-34-234-2C
rancher	i-0a2d7321848f80501	Running	t2.large	2/2 checks passed	No alarms	us-east-1a	ec2-52-70-84-

9. Now, we need to update the instance using the command - **sudo apt-get update**
10. Then we need to install Docker using the command - **sudo apt install docker.io**
11. Verify that docker is installed by running the command - **docker -v**
12. Then we need to run the docker command to install rancher - **sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher**
13. After waiting for some time we need to open the public IPv4 address and we will be redirected to the rancher page. We will be given few instructions to get our password. We can set our own password after logging in.

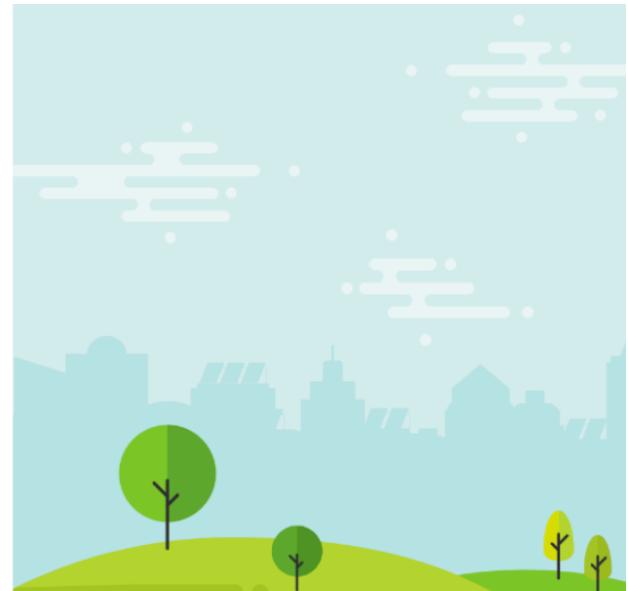
The screenshot shows the Rancher login page. The header says "Howdy! Welcome to Rancher". A message states: "It looks like this is your first time visiting Rancher: If you pre-set your own bootstrap password, enter it here. Otherwise a random one has been generated for you. To find it:" Below this are two bullet points for a "docker run" installation:

- Find your container ID with `docker ps`, then run:
- `docker logs container_id 2>&1 | grep "Bootstrap Password:"`

For a Helm installation, run:

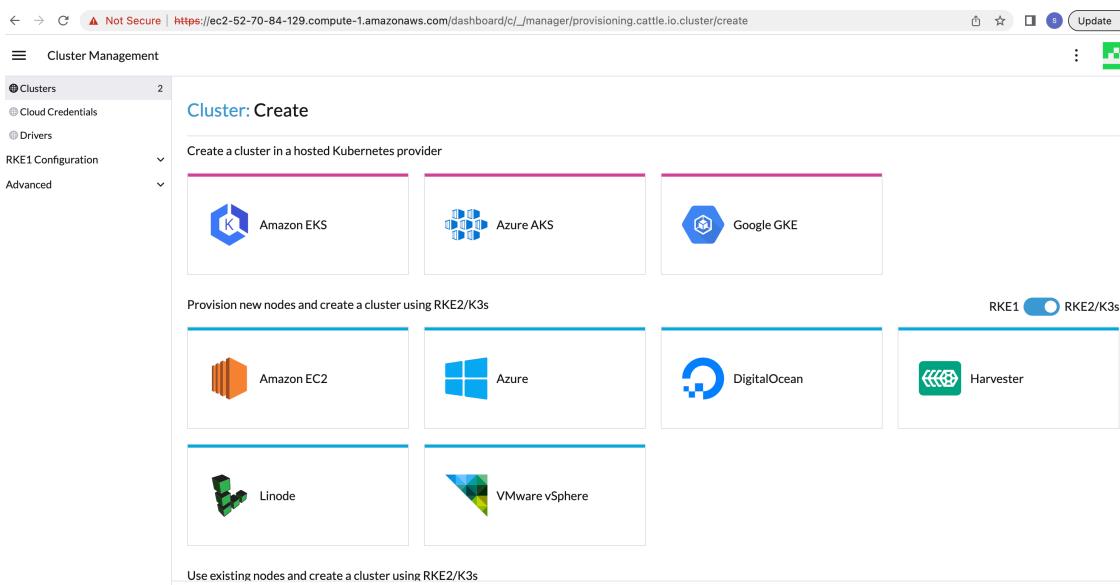
```
kubectl get secret --namespace cattle-system bootstrap-secret -o go-template="{{.data.bootstrapPassword|base64decode}}{{"\n"}}
```

Below the instructions is a password input field with a "Show" link and a "Log in with Local User" button.



PART 5 - Create a cluster using rancher:

1. After we come up with the rancher page we need to login into rancher and for that we need to set up our own password.
2. We need to run ‘sudo docker ps’ command to get the container id and use the command in the above image to extract the Bootstrap password and then login successfully using our password.
3. After login we can see that a local cluster is already available in homepage.
4. Then we need to click on Create new cluster and choose Google GKE as an option.



5. After that give a unique name to the cluster and assign a project ID to it using GCP - <https://console.cloud.google.com/> and also create a Service Account using GCP and download the json file to create cluster.

Not Secure | https://ec2-52-70-84-129.compute-1.amazonaws.com/dashboard/c/_/manager/provisioning.cattle.io.cluster/create?type=googleke

Cluster Management

Add Cluster - Google GKE

Clusters 2

Cloud Credentials

Drivers

RKE1 Configuration

Advanced

Cluster Name *

Add a Description

Member Roles

Labels & Annotations

None

Expand All

Google Account Access

Choose the Google Project ID and Cloud Credential that will be used to launch your cluster.

Google Project ID *

Cloud Credentials

ServiceAccount

Add New

Cancel Next: Configure Cluster

6. Create a project in GCP using free access and navigate to Kubernetes cluster and enable them.

console.cloud.google.com/welcome?project=swe645-assignment2-384718&authuser=2

Free trial status: \$279.19 credit and 88 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Google Cloud SWE645-Assignment2 Search

Select a project NEW PROJECT

Search projects and folders

RECENT STARRED ALL

Name ID

You're working in SWE645-Assignment2 ?

swe645-assignment2-384718

Kubernetes Engine

Kubernetes clusters

Containers package an application so it can easily be deployed to run in its own isolated environment. Containers are run on Kubernetes clusters. [Learn more](#)

[CREATE](#) [DEPLOY CONTAINER](#) [TAKE THE QUICKSTART](#)

7. Now create a Service Account User required to create a cluster like this and give it a unique name and we need to assign roles like – Viewer, Kubernetes Engine admin, Compute Viewer and Service Account User and then click continue and done.

The screenshot shows the Google Cloud IAM & Admin Service accounts page. On the left, there's a sidebar with options like IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (which is selected), Workload Identity Federat..., Labels, Tags, Settings, and Privacy & Security. The main area displays a table of service accounts for project "SWE645-Assignment2". The table has columns for Email, Status, Name, Description, Key ID, and Key creation date. Two entries are listed:

Email	Status	Name	Description	Key ID	Key creation date
668801257316-compute@developer.gserviceaccount.com	✓	Compute Engine default service account	No keys		
rancher-swe645-2@swe645-assignment2-384718.iam.gserviceaccount.com	✓	rancher/swe645-2		5a2b3041375a3f486a781ae0ff3cc51445d2bcb	Apr 24, 2023

The screenshot shows the "Create service account" form. On the left, there's a sidebar with the same navigation options as the previous page. The main area is titled "Create service account" and has a step indicator "1 Service account details". It contains fields for "Service account name" (set to "service1/rancher"), "Service account ID" (set to "service1-rancher"), and "Email address" (set to "service1-rancher@swe645-assignment2-384718.iam.gserviceaccount.com"). Below these are "Service account description" and "CREATE AND CONTINUE" buttons.

Free trial status: \$279.19 credit and 88 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Create service account

Grant this service account access to project (optional)

Grant this service account access to SWE645-Assignment2 so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role	IAM condition (optional)	Action
Compute Viewer	+ ADD IAM CONDITION	<input type="button" value="Delete"/>
Kubernetes Engine Admin	+ ADD IAM CONDITION	<input type="button" value="Delete"/>
Service Account User	+ ADD IAM CONDITION	<input type="button" value="Delete"/>
Viewer	+ ADD IAM CONDITION	<input type="button" value="Delete"/>

Service account created

- Now we can see that the service account got created and now we need to click on Manage keys from the three dot icon beside the service.

Service accounts

Service accounts for project "SWE645-Assignment2"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts](#)

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies](#)

Filter	Email	Status	Name	Description	Key ID	Key creation date	Actions
<input type="checkbox"/>	compute@developer.gserviceaccount.com	✓	Compute Engine default service account	No keys		0	<input type="button" value="More"/>
<input type="checkbox"/>	rancher@swe645-assignment2-384718.iam.gserviceaccount.com	✓	rancher@swe645-2	5a2b3041375a3f486a781ae0f9f3cc51445d2bcb	Apr 24, 2023	1	<input type="button" value="More"/>

- Now click on Add key and create a JSON key.

The screenshot shows the Rancher interface for managing a cluster named "rancher/swe645-2". The "KEYS" tab is selected. A modal dialog titled "Create private key for 'rancher/swe645-2'" is open, prompting the user to download a private key file. The "Key type" is set to "JSON" (selected radio button). Below it, "P12" is also listed as an option. The "CREATE" button at the bottom right of the modal is highlighted in blue.

The screenshot shows the Google Cloud IAM & Admin interface. The left sidebar is under "IAM & Admin" and has "Service Accounts" selected. The main area shows the "KEYS" tab for a service account named "rancher/swe645-2". A table lists one key entry:

Type	Status	Key	Key creation date	Key expiration date
	Active	5a2b3041375a3f486a781ae0f9f3cc51445d2bcb	Apr 24, 2023	Dec 31, 9999

- Now add the service account to the cluster in rancher by selecting “Read from a file” in rancher and select the downloaded JSON file from our local machine and click on next.
- Then choose and confirm the local zones and click on Configure cluster.

The screenshot shows the Cluster Management interface with the following configuration details:

- Member Roles:** Control who has access to the cluster and what permission they have to change it.
- Labels & Annotations:** Configure labels and annotations for the cluster. Status: None.
- Google Account Access:** Choose the Google Project ID and Cloud Credential that will be used to launch your cluster.
 - Google Project ID: swe645-assignment2-384718
 - Cloud Credential: us-central1-c
- Cluster Location:** Configure cluster and node location options.
 - Location Type: Zonal (selected)
 - Zone: us-central1-c
 - Additional Zones: us-central1-a, us-central1-b, us-central1-f

Buttons at the bottom: Cancel and Configure Cluster.

12. Then under Node pools, name the node pool as “worker” and set the initial count to 3 and leave the remaining settings as default.

The screenshot shows the Node Pools configuration interface with the following details:

- Node Labels:** + Add Label
- Network Tags:** + Add Tag
- Group Details:**

Name *	workers	Initial Node Count	3	Max Pod Per Node	110
--------	---------	--------------------	---	------------------	-----
- Autoscaling:** Autoscaling (unchecked)
- Auto Repair:** Auto Repair (checked)
- Auto Upgrade:** Auto Upgrade (checked)

13. It will take 5-10 min to create a cluster from provisioning state and in case of error view the logs and create it again.

Welcome to Rancher

You can change what you see when you login via preferences

Clusters 2

State	Name	Provider	Kubernetes Version	CPU	Memory	Pods
Active	cluster	Google GKE	v1.25.7-gke.1000	23.73 cores	78 GiB	29/330
Active	local	Local K3s	v1.25.5+k3s1	2 cores	7.76 GiB	6/110

Manage Import Existing Create Filter

Links

- Docs
- Forums
- Slack
- File an Issue
- Get Started
- Commercial Support

What's new in 2.7

PART 6 – DEPLOYING DOCKER IMAGE IN RANCHER/ CLUSTER

- After creation of cluster, navigate to the hamburger menu in rancher and select the cluster that you created.

← → ⌂ Not Secure | https://

☰ RANCHER®

[Home](#)

EXPLORE CLUSTER

- [cluster](#)
- [local](#)

GLOBAL APPS

- [Continuous Delivery](#)
- [Cluster Management](#)
- [Virtualization Management](#)

CONFIGURATION

- [Users & Authentication](#)
- [Extensions](#)
- [Global Settings](#)

- Select the cluster from the hamburger menu and navigate to “Projects/namespace” in rancher.

- Now create a project and also a namespace under that project.

The screenshot shows the 'Projects/Namespace' list in the OpenShift web console. On the left is a sidebar with 'Cluster' and 'Workloads' sections. The main area lists namespaces:

- Project: Default**: Contains the namespace **default**. A 'Create Namespace' button is available.
- Project: homework2**: Contains the namespace **namespace-h2**. A 'Create Namespace' button is available.
- Not in a Project**: Contains the namespace **local**. A 'Create Namespace' button is available.
- Project: System**: No namespaces are defined.

A message at the bottom states: "There are no namespaces defined."

- Now navigate to workloads tab and click on deployments to create a new deployment.
- Add a new deployment name and select the namespace that we created and add 3 replicas to the deployment.
- Then add the container image name to the deployment and add port or service.

The screenshot shows the 'Deployment: Create' form in the OpenShift web console. The sidebar shows 'Workloads' with 1 deployment listed. The main form fields are:

- Namespace**: namespace-h2
- Name**: deployment-a2
- Replicas**: 3

The 'General' tab is selected, showing:

- General** section: Container Name is set to **container-0**, and the type is **Standard Container**.
- Image** section: Container Image is **sanjanagovindu/swe645-a2**, and Pull Policy is **Always**.

At the bottom are 'Cancel', 'Edit as YAML', and 'Create' buttons.

7. Then under the ‘Networking’ tab add the Service Type as ‘Load Balancer’ and give it a name and add the private container port and listening port as 8080 by leaving the protocol as TCP.

The screenshot shows the AWS Lambda console interface for creating a new deployment. The left sidebar lists resources: Cluster, Workloads (CronJobs, DaemonSets, Deployments, Jobs, StatefulSets, Pods), Apps, Service Discovery, Storage, Policy, and More Resources. The main area is titled 'Deployment' and contains tabs for 'Image', 'Storage', 'Networking', 'Command', and 'Environment Variables'. In the 'Networking' tab, the 'Service Type' is set to 'Load Balancer', the 'Name' is 'loadbalancer', the 'Private Container Port' is '8080', and the 'Listening Port' is '8080'. Buttons at the bottom include 'Cluster Tools', 'v2.7.3', 'Cancel', 'Edit as YAML', and a large blue 'Create' button.

8. Leave the Deployment and Pod settings to default and create a deployment. It will usually take 5 minutes time to get created.

The screenshot shows the AWS Lambda console interface displaying the list of existing deployments. The left sidebar is identical to the previous screenshot. The main area is titled 'Deployments' and shows a table with the following data:

	State	Name	Image	Ready	Up To Date	Available	Restarts	Age	Health
<input type="checkbox"/>	Active	deployment-a2	sanjanagovindu/swe645-a2	3/3	3	3	0	4.1 hours	<div style="width: 100%; background-color: #2e7131;"></div>

Buttons at the top of the table include 'Redeploy', 'Download YAML', and 'Delete'. A 'Create' button is located in the top right corner of the main area.

Deployment: deployment-a2 (Active)

Namespace: namespace-h2 Age: 4.1 hours Pod Restarts: 0

Image: sanjanagovindu/swe645-a2 Ready: 3/3 Up-to-date: 3 Available: 3

Endpoints: 8080/TCP

Annotations: Show 1 annotation

Pods by State

State	Name	Image	Ready	Restarts	IP	Node	Age
Running	deployment-a2-7f8579777c-98lbp	sanjanagovindu/swe645-a2	1/1	0	10.0.1.11	gke-cluster-nodes-94180deb-3qtb	3.7 hours
Running	deployment-a2-7f8579777c-hmttj	sanjanagovindu/swe645-a2	1/1	0	10.0.0.16	gke-cluster-nodes-94180deb-bxk5	3.7 hours
Running	deployment-a2-7f8579777c-zqqxf	sanjanagovindu/swe645-a2	1/1	0	10.0.2.13	gke-cluster-nodes-94180deb-trrj	3.7 hours

9. Confirm if the deployment works by selecting the deployment-a2 service or by clicking on “8080/TCP” endpoint and add your WAR file name to the link to access your survey form.

Service	Type	Port	Endpoint	Cluster IP	Age
deployment-a2	8080/TCP	workload.user.cattle.io/workloadselector=apps.deployment-namespace-h2-deployment-a2	Cluster IP	4.2 hours	
deployment-a2-loadbalancer	8080/TCP	workload.user.cattle.io/workloadselector=apps.deployment-namespace-h2-deployment-a2	Load Balancer	4.2 hours	

DEPARTMENT OF COMPUTER SCIENCE

GMU CS DEPARTMENT - FEEDBACK

USERNAME*:

STREET ADDRESS*:

CITY*:

STATE*:

ZIP CODE*:

TELEPHONE NUMBER*: (###-###-###)

E-MAIL*: johnsmith@gmail.com

URL*: <https://www.github.com/john-smith/>

DATE OF SURVEY*: MM-DD-YYYY

PART 7 – INSTALL AND SETUP JENKINS

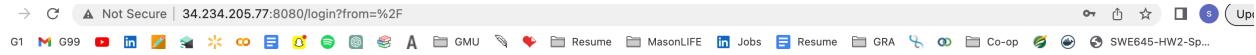
1. For setting up Jenkins we need to create another EC2 instance in AWS Lab
2. EC2 > Launch Instance > Give a name > Select Ubuntu > Add the existing pem file > Allow traffic from HTTP and HTTPS and configure volume to 50 > launch instance (basically follow the same steps in Part 3 as we did for rancher)

The screenshot shows the AWS EC2 Instances details page for an instance named 'jenkins'. The instance ID is i-0a6221165b86f893c. The public IPv4 address is 34.234.205.77. The instance state is Running. Other details include the private IP address 172.31.91.252, the public IPv4 DNS name ec2-34-234-205-77.compute-1.amazonaws.com, and the elastic IP address 34.234.205.77. The instance type is t2.medium, and it is associated with a VPC ID vpc-0fdb3f8413c346f6a and a subnet ID subnet-01b60b0b4a676347f.

The screenshot shows the AWS EC2 Instances list page. There are two instances listed: 'jenkins' (instance ID i-0a6221165b86f893c) and 'rancher' (instance ID i-0a2d7321848f80501). Both instances are in the 'Running' state. The 'jenkins' instance is associated with a t2.medium instance type, while the 'rancher' instance is associated with a t2.large instance type. They are both located in the us-east-1d availability zone and have the public IPv4 DNS names ec2-34-234-205-205.compute-1.amazonaws.com and ec2-52-70-84-1 respectively.

3. Then after instance is launched install docker and add the jenkins user to the docker group using the command which enables using docker commands without using sudo – “sudo usermod -aG docker jenkins”

4. Install JDK 11 on EC2 instance in AWS itself by clicking on SSH connect and run this command if the key is not publicly viewable - chmod 400 swe645-assign2.pem and for jdk use - sudo apt install openjdk-11-jdk” and check the java version installed.
5. Use <https://www.jenkins.io/doc/book/installing/linux/#long-term-support-release> to install Jenkins on local machine and follow the steps.
6. Now we need to verify if Jenkins is installed in our local machine running “systemctl status jenkins”
7. To enable continuous deployment through Jenkins we need to install kubectl using commands - “sudo apt install snapd” and “sudo snap install kubectl --classic”
8. We need to go to rancher again and download the cluster configuration for Jenkins and switch to Jenkins users using “sudo su jenkins”
9. We need to create a file using vi editor and paste the downloaded cluster configuration YAML which is “\$HOME/.kube/config”
10. We need to verify if kubectl and the configuration is working correctly using command “kubectl config current-context”
11. Output will be the name of the cluster if verified correctly.
12. We need to use the configured elastic IP within the port 8080 to access Jenkins from the browser and follow the steps to setup user name and password for Jenkins.



13. Then we need to setup the credentials for docker and github by navigating to Manage Jenkins > Manage Plugins > Install the docker, git, githib, pipeline plugins then Manage Jenkins > Manage Credentials to add the credentials.

The screenshot shows the Jenkins Plugin Manager interface. On the left, there's a sidebar with links: 'Updates', 'Available plugins' (selected), 'Installed plugins', and 'Advanced settings'. The main area is titled 'Plugins' and has a search bar containing 'docker'. Below the search bar, a table lists three Docker-related plugins:

Name	Enabled
Docker API Plugin	<input checked="" type="checkbox"/>
Docker Commons Plugin	<input checked="" type="checkbox"/>
Docker Pipeline	<input checked="" type="checkbox"/>

Each plugin entry includes a brief description and a 'Report an issue with this plugin' link. A message at the bottom of the list states: 'This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.'

The screenshot shows the Jenkins Credentials page. The sidebar on the left has links: 'Dashboard', 'Manage Jenkins' (selected), and 'Credentials'. The main area is titled 'Credentials' and displays a table of stored credentials:

T	P	Store	Domain	ID	Name
		System	(global)	ubuntu	ubuntu (ubuntu)
		System	(global)	gsanjana8989@gmail.com	github
		System	(global)	Github	sanjana-govindu*****
		System	(global)	docker	sanjanagovindu***** (docker)

Below the table, a section titled 'Stores scoped to Jenkins' shows a single entry:

P	Store	Domains
	System	(global)

14. Then we go to dashboard and click on ‘New Item’ to create a pipeline and name the pipeline as a2 for this assignment.



Dashboard >

+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

rancher-node

1 Idle

A screenshot of the Jenkins dashboard showing the search bar with "Search (⌘+K)" and a user profile for "Sanjana". Below the search bar, the breadcrumb navigation shows "Dashboard > All >".

A modal dialog box titled "Enter an item name" with a red error message "» This field cannot be empty, please enter a valid name". Below the input field, there are three project types listed: "Freestyle project", "Pipeline", and "Multi-configuration project", each with a brief description and icon.

15. In the configuration, set the “Build Triggers” by enabling the “Poll SCM” and add “*****” so that the repository can be checked in every minute.

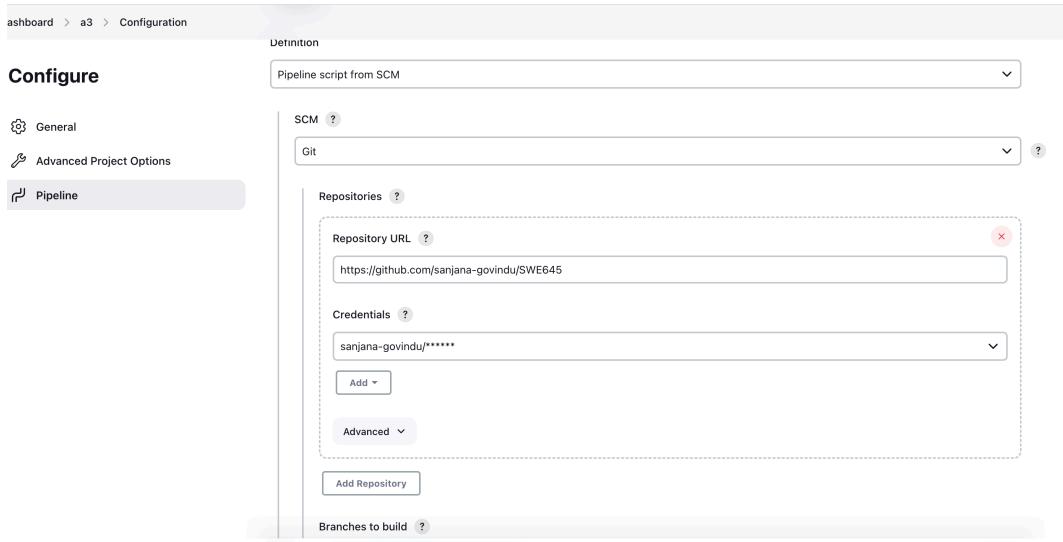
The screenshot shows the Jenkins 'Configure' screen for a project named 'a3'. Under the 'General' tab, there are sections for 'Build Triggers' and 'Pipeline'. In the 'Build Triggers' section, the 'Poll SCM' option is selected (indicated by a checked checkbox). Below it, there is a 'Schedule' field containing the cron expression '*****'. Other trigger options like 'Build after other projects are built', 'Build periodically', and 'GitHub hook trigger for GITScm polling' are also listed but not selected.

16. Scroll down and in Pipeline set it to “Pipeline script from SCM” and select Git > Provide the Github repository URL and for GitHub personal access token should be used instead of the password and it must have read and write access to access the GitHub repository.

The screenshot shows the GitHub 'Personal access tokens (classic)' settings page. On the left, there are tabs for 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens'. The 'Personal access tokens' tab is active, showing a sub-tab for 'Tokens (classic)'. On the right, a table lists a single token:

Personal access tokens (classic)		Generate new token	Revoke
jenkins	— admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete:packages, gist, notifications, project, repo, user, workflow, write:discussion, write:packages	Never used	Delete
Expires on Sun, Jul 23 2023.			

A note at the bottom explains that personal access tokens function like ordinary OAuth access tokens and can be used instead of a password for Git over HTTPS, or to authenticate to the API over Basic Authentication.



17. Leave the script path in our Jenkins file in the root directory and then uncheck the Lightweight checkout.



18. Leave everything else to default and create the pipeline.

19. The Jenkins File in Github is like this:

```

main SWE645 / Jenkinsfile Go to file ..
sanjana-govindu Update Jenkinsfile Latest commit fc150ac 16 hours ago History
All 1 contributor
39 lines (39 sloc) | 959 Bytes Raw Blame ⌂ ⌃ ⌄ ⌅ ⌆ ⌇
1 pipeline {
2   agent {
3     node {
4       label 'swe-645-a2'
5     }
6   }
7   stages {
8     stage('Checkout') {
9       steps {
10         checkout scm
11     }
12   }
13   stage('Build') {
14     steps {
15       sh 'rm -rf *.war'
16       sh 'jar -cvf SWE645-Assignment2.war -C WebContents/'
17     }
18   post {
19     success {
20       archiveArtifacts artifacts: '*.war', fingerprint: true
21     }
22   }
23 }
24 stage('Docker Build and Push') {
25   steps {
26     withCredentials([usernamePassword(credentialsId: 'docker', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')) {

```

It retrieves the configured credentials of docker and in first stage the repository is cloned and previous WAR files are removed and new WAR is generated along with a new build of the docker image and then the docker image is pushed to docker hub and it verifies if the dockerhub credentials are correct here. Then the deployment is restarted and it will check a d pull image from docker hub with latest updates and we can observe the deployment in rancher and can see that the pods are restarted. Finally the pipeline is created here and everything is shown as green. We can see the final pipeline in Jenkins and a check if everything is configured correctly. The pipeline will be built and triggered and chanced will be deployed and we can see a version change in the webpage. If any errors are encountered we need to check the logs and correct them.

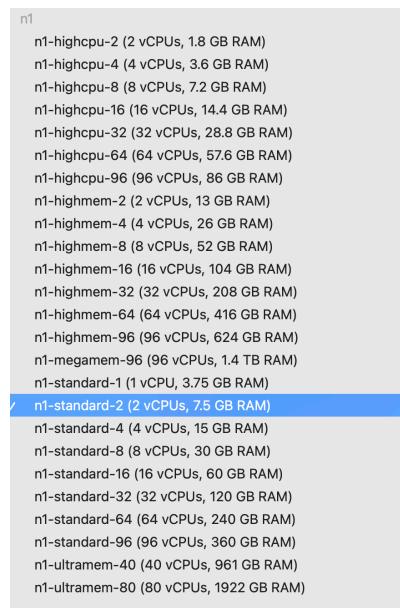
STAGE VIEW:

Stage	Average Time
Declarative: Checkout SCM	1s
Checkout	389ms
Build	1s
Docker Build and Push	3s
Deploy to Kubernetes	431ms

CHALLENGES FACED DURING THIS ASSIGNMENT/ LESSONS LEARNED:

- In this assignment we faced a major issue while pushing the image to docker hub and running it locally. We are unable to build the docker image locally properly if we use FROM tomcat or FROM tomcat-10.4-jdk11 in DockerFile so we need to use FROM amd64/tomcat for sure if we are using MAC OS.
- While creating an EC2 instance, we were facing an issue if the instance type was set to t2.medium so we used t2.large and we also need to configure the volume to greater than 30 here we used 50. If its set to default the instance is not working properly for later use.
- And while creating cluster we need to set the node pools machine type to a higher version so that we don't face an issue and the cluster can handle it.

The screenshot shows the 'Cluster Management' interface. On the left, there's a sidebar with options: Clusters (selected), Cloud Credentials, Drivers, RKE1 Configuration, and Advanced. Under Clusters, there are two clusters listed. The main area is titled 'Node Pools' with a sub-section 'Customize node pools'. It shows 'Node Details' for a node pool. The Kubernetes Version is 1.25.7-gke.1000. The Image Type is 'Container-Optimized OS with Containerd'. The Machine Type is set to 'n1-standard-2 (2 vCPUs, 7.5 GB RAM)'. The Root disk type is 'Standard persistent disk'. The Root Disk Size is 100 GB. Local SSD disks are set to 0 GB. There's a checkbox for 'Preemptible nodes (beta)' which is unchecked. Below that is a section for 'Taints' with a '+ Add Taint' button. At the top right, there's a 'Remove Node Pool' link and a green checkmark icon.



- So if we don't set the machine type to a higher version we can't create a deployment and it shows cashback loop errors where the pods fail and we need to start everything from the beginning and clear cache. Below are the deployment errors. So we need to choose the machine type carefully so that it will be suitable for our assignment.
- In my case if we use a very high-end machine the google cloud platform doesn't have enough memory to handle the deployment and then it will show quota is not enough for a free account so we had to try creating the cluster multiple times to handle our deployment.

Deployments ☆

[⟳ Redeploy](#) [⬇️ Download YAML](#) [trash Delete](#)

<input type="checkbox"/> State	Name	Image	Ready	Up
Namespace: swe645-2				
<input type="checkbox"/> Updating	swe645	sanjanagovindu/swe645-hw2	0/3	3
Deployment does not have minimum availability.				

Deployment does not have minimum availability.

image: sanjanagovindu/swe645-hw2 Ready: 0/3 Up-to-date: 3 Available: 0 endpoints: 8080/TCP annotations: Show 1 annotation

[ods by State](#) [Scale - 3](#)

<input type="checkbox"/> Crashloopbackoff
3

[Pods](#) [Services](#) [Ingresses](#) [Conditions](#) [Recent Events](#) [Related Resources](#)

[⬇️ Download YAML](#) [trash Delete](#)

<input type="checkbox"/> State	Name	Image	Ready	Restarts	IP	Node	Age
<input type="checkbox"/> Crashloopback...	swe645-55c7bb97f9-r5Bbp	sanjanagovindu/swe645-hw2	0/1	3 (33s ago)	10.56.2.10	gke-swe645-workers-53e8927b-5pwz	1.4 mins
Containers with unready status: [container-0]							
<input type="checkbox"/> Crashloopback...	swe645-55c7bb97f9-rkvz5	sanjanagovindu/swe645-hw2	0/1	3 (31s ago)	10.56.1.13	gke-swe645-workers-53e8927b-9xxz	1.4 mins
Containers with unready status: [container-0]							
<input type="checkbox"/> Crashloopback...	swe645-55c7bb97f9-s28qx	sanjanagovindu/swe645-hw2	0/1	3 (34s ago)	10.56.0.11	gke-swe645-workers-53e8927b-x0pz	1.4 mins
Containers with unready status: [container-0]							

Deployment does not have minimum availability.

Image: sanjanagovindu/swe645-hw2 Ready: 0/3 Up-to-date: 3 Available: 0
Endpoints: 8080/TCP
Annotations: Show 1 annotation

Pods by State

State	Name	Image	Ready	Restarts	IP	Node	Age
Error	swe645-55c7bb97f9-r58bp	sanjanagovindu/swe645-hw2	0/1	4 (49s ago)	10.56.2.10	gke-swe645-workers-53e8927b-5pwz	1.7 mins
Error	swe645-55c7bb97f9-rkvz5	sanjanagovindu/swe645-hw2	0/1	4 (52s ago)	10.56.1.13	gke-swe645-workers-53e8927b-9xxz	1.7 mins
Crashloopback...	swe645-55c7bb97f9-s28qx	sanjanagovindu/swe645-hw2	0/1	4 (11s ago)	10.56.0.11	gke-swe645-workers-53e8927b-x0pz	1.7 mins

Role of team members in this assignment:

- **ADITEE:** Worked on containerizing the web application using docker and then pushed the image into Docker Hub to make it globally accessible for further use.
- **SANJANA:** Worked on setting up Rancher and creating a cluster using rancher. Then deployed the docker image using docker. Made it work using the Kubernetes cluster so that it can be made available on internet. We created a cluster using Kubernetes installed and pods and nodes running and linked it to our docker image to make it available to the internet. Also worked on setting up Jenkins using rancher. Built the pipeline in Jenkins.

References:

- Referred the setup instructions uploaded in blackboard by the professor.
Install docker desktop on Mac: <https://docs.docker.com/desktop/install/mac-install/>
- <https://docs.docker.com/engine/install/linux-postinstall/>
- Rancher AWS Quick Start Guide: <https://rancher.com/docs/rancher/v2.5/en/quick-start-guide/deployment/amazon-aws-qs/>
- Deployment using Kubernetes:
<https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
- <https://www.jenkins.io/doc/book/pipeline/docker/>