Docker

Projects
- Hello World - Java, JavaScript and Python
- 2 Microservices - Currency Exchange and Currency Conversion

Steps
- Step 01 - Docker and DevOps - Installation and Introduction
- Step 02 - Your First Docker Usecase
- Step 03 - Important Docker Concepts - Registry, Repository, Tag, Image and Container
- Step 04 - Playing with Docker Images - Java, JavaScript and Python
- Step 05 - Playing with Docker - Detached Mode and Logs
- Step 06 - Playing with Docker Images and Containers
- Step 07 - Understanding Docker Architecture - Docker Client, Docker Engine
- Step 08 - Understanding Docker Popularity - My 3 Top Reasons
- Step 09 - Learning Docker Images - Commands
- Step 10 - Learning Docker Containers - Commands
- Step 11 - Learning Docker Commands - system and stats
- Step 12 - Building Docker Images for Python Application
- Step 13 - Understanding creation of Docker Images in Depth
- Step 14 - Pushing Python App Docker Image to Docker Hub
- Step 15 - Building and Pushing Docker Image for Node JavaScript App
- Step 16 - Building and Pushing Docker Image for Java Application
- Step 17 - Building Efficient Docker Images - Improving Layer Caching
- Step 18 - Understanding ENTRYPOINT vs CMD
- Step 19 - Docker and Microservices - Quick Start
- Step 20 - Introduction to Microservices - CE and CC
- Step 21 - Running Microservices as Docker Containers
- Step 22 - Using Docker Link to Connect Microservices
- Step 23 - Using Custom Networking to Connect Microservices
- Step 24 - Using Docker Compose to Simplify Microservices Launch
- Step 25 - Understanding Docker Compose further

Registry and Repositories

- https://hub.docker.com/u/in28min
- https://hub.docker.com/r/in28min/hello-world-java
- https://hub.docker.com/r/in28min/hello-world-python
- https://hub.docker.com/r/in28min/hello-world-nodejs

Commands

```bash
# Check the version of Docker installed
docker --version

# Run a Docker container with a Python hello world application
docker run -p 5000:5000 in28min/hello-world-python:0.0.1.RELEASE

# Run a Docker container with a Java hello world application
docker run -p 5000:5000 in28min/hello-world-java:0.0.1.RELEASE

# Run a Docker container with a Node.js hello world application
docker run -p 5000:5000 in28min/hello-world-nodejs:0.0.1.RELEASE

# Run a Docker container with a Node.js hello world application in detached mode
docker run -d -p 5000:5000 in28min/hello-world-nodejs:0.0.1.RELEASE

# Run a Docker container with a Python hello world application in detached mode
docker run -d -p 5001:5000 in28min/hello-world-python:0.0.1.RELEASE
```

```
# View the logs of a Docker container
docker logs 04e52ff9270f5810eefe1f77222852dc1461c22440d4ecd6228b5c38f09d838e

# View the logs of a Docker container with a given container ID or name
docker logs c2ba

# List Docker images
docker images

# List running containers
docker container ls

# List all containers, including stopped ones
docker container ls -a

# Stop a running container with a given container ID
docker container stop f708b7ee1a8b

# Run a Docker container with a REST API in detached mode
docker run -d -p 5001:8080 in28min/hello-world-rest-api:0.0.1.RELEASE

# Pull a Docker image from a registry
docker pull mysql

# Search for Docker images in a registry
docker search mysql

# Show the history of a Docker image
docker image history in28min/hello-world-java:0.0.1.RELEASE

# Show the history of a Docker image with a given image ID
docker image history 100229ba687e

# Inspect a Docker image
docker image inspect 100229ba687e

# Remove a Docker image
docker image remove mysql

# Remove a Docker image with a given image name and tag
docker image remove in28min/hello-world-java:0.0.1.RELEASE

# Remove a Docker container with a given container ID or name
docker container rm 3e657ae9bd16

# List all containers, including stopped ones
docker container ls -a

# Pause a running container with a given container ID or name
docker container pause 832

# Unpause a paused container with a given container ID or name
docker container unpause 832

# Stop a running container with a given container ID or name
docker container stop 832

# Inspect a Docker container with a given container ID or name
docker container inspect ff521fa58db3

# Remove all stopped containers
docker container prune

# Show Docker system information
```

```
docker system

# Show Docker disk usage
docker system df

# Show detailed Docker system information
docker system info

# Remove all unused Docker resources
docker system prune -a

# Display the top resource-consuming processes of a Docker container with a
given container ID or name
docker top 9009722eac4d

# Show live resource usage statistics of a Docker container with a given
container ID or name
docker stats 9009722eac4d

# Run a Docker container with a Java hello world application, limited to 512MB
memory
docker container run -p 5000:5000 -d -m 512m in28min/hello-world-
java:0.0.1.RELEASE

# Run a Docker container with a Java hello world application, limited to 512MB
memory and CPU quota of 50%
docker container run -p 5000:5000 -d -m 512m --cpu-quota=50000 in28min/hello-
world-java:0.0.1.RELEASE

# Show Docker system events
docker system events

# Show live resource usage statistics of a Docker container with a given
container ID or name
docker container stats
4faca1ea914e3e4587d1d790948ec6cb8fa34f26e900c12632fd64d4722fd59a

# Show live resource usage statistics of a Docker container with a given
container ID or name
docker stats 42f170966ce613d2a16d7404495af7b3295e01aeb9142e1fa1762bbdc581f502

# Change directory to the Python hello world project
cd /in28Minutes/git/devops-master-class/projects/hello-world/hello-world-python

# Build a Docker image with a given tag
docker build -t in28min/hello-world-python:0.0.2.RELEASE .

# Run a Docker container with a Python hello world application, mapped to port
5000
docker run -p 5000:5000 -d in28min/hello-world-python:0.0.2.RELEASE

# Show the history of a Docker image with a given image ID
docker history e66dc383f7a0

# Push a Docker image to a registry
docker push in28min/hello-world-python:0.0.2.RELEASE

# Change directory to the Node.js hello world project
cd ../hello-world-nodejs/

# Build a Docker image with a given tag
docker build -t in28min/hello-world-nodejs:0.0.2.RELEASE .
```

```
# Run a Docker container with a Node.js hello world application, mapped to port
5000
docker container run -d -p 5000:5000 in28min/hello-world-nodejs:0.0.2.RELEASE

# Push a Docker image to a registry
docker push in28min/hello-world-nodejs:0.0.2.RELEASE

# Change directory to the Java hello world project
cd ../hello-world-java/

# Build a Docker image with a given tag
docker build -t in28min/hello-world-java:0.0.2.RELEASE .

# Run a Docker container with a Java hello world application, mapped to port
5000
docker run -d -p 5000:5000 in28min/hello-world-java:0.0.2.RELEASE

# Push a Docker image to a registry
docker push in28min/hello-world-java:0.0.2.RELEASE

# Run a Docker container with a Node.js hello world application, mapped to port
5001 and ping google.com
docker run -d -p 5001:5000 in28min/hello-world-nodejs:0.0.3.RELEASE ping
google.com

# Run a Docker container with a currency exchange service
docker run -d -p 8000:8000 --name=currency-exchange in28min/currency-
exchange:0.0.1-RELEASE

# Run a Docker container with a currency conversion service
docker run -d -p 8100:8100 --name=currency-conversion in28min/currency-
conversion:0.0.1-RELEASE

# List Docker networks
docker network ls

# Inspect a Docker network with a given network name
docker network inspect bridge

# Run a Docker container with a currency conversion service, linked to the
currency exchange service and environment variable set
docker run -d -p 8100:8100 --env CURRENCY_EXCHANGE_SERVICE_HOST=http://currency-
exchange --name=currency-conversion --link currency-exchange in28min/currency-
conversion:0.0.1-RELEASE

# Create a Docker network with a given network name
docker network create currency-network

# Stop a running container with a given container ID or name
docker container stop currency-exchange

# Stop a running container with a given container ID or name
docker container stop currency-conversion

# Run a Docker container with a currency exchange service, connected to the
currency-network
docker run -d -p 8000:8000 --name=currency-exchange --network=currency-network
in28min/currency-exchange:0.0.1-RELEASE

# Run a Docker container with a currency conversion service, connected to the
currency-network and environment variable set
docker run -d -p 8100:8100 --env CURRENCY_EXCHANGE_SERVICE_HOST=http://currency-
exchange --name=currency-conversion --network=currency-network in28min/currency-
conversion:0.0.1-RELEASE
```

```
# Check the version of Docker Compose installed
docker-compose --version

# Change directory to the microservices folder
cd ../../microservices/

# Start the services defined in the Docker Compose file
docker-compose up

# Start the services defined in the Docker Compose file in detached mode
docker-compose up -d

# List running containers
docker container ls

# List Docker networks
docker network ls

# Inspect a Docker network with a given network name
docker network inspect microservices_currency-compose-network

# Stop the services defined in the Docker Compose file
docker-compose down

# List all containers, including stopped ones
docker container ls -a

# Remove all unused Docker resources
docker system prune -a

# Validate the Docker Compose file
docker-compose config

# List Docker images used by the services defined in the Docker Compose file
docker-compose images

# List the running services defined in the Docker Compose file
docker-compose ps

# Display the top resource-consuming processes of the services defined in the
Docker Compose file
docker-compose top


```


```bash
# Build a Docker image for a Java hello world application with a given tag
docker build -t in28min/hello-world-java:0.0.1.RELEASE .

# Push the Docker image with the specified tag to a registry
docker push in28min/hello-world-java:0.0.1.RELEASE

# Build a Docker image for a Python hello world application with a given tag
docker build -t in28min/hello-world-python:0.0.1.RELEASE .

# Push the Docker image with the specified tag to a registry
docker push in28min/hello-world-python:0.0.1.RELEASE
```

```
# Build a Docker image for a Node.js hello world application with a given tag
docker build -t in28min/hello-world-nodejs:0.0.1.RELEASE .

# Push the Docker image with the specified tag to a registry
docker push in28min/hello-world-nodejs:0.0.1.RELEASE

```

Host Networking in Docker for Mac and Windows

- https://docs.docker.com/network/host/

>The host networking driver only works on Linux hosts, and is not supported on Docker Desktop for Mac, Docker Desktop for Windows, or Docker EE for Windows Server.

Kubernetes

Projects
- Hello World REST API
- 2 Microservices - Currency Exchange and Currency Conversion

Steps
- Step 01 - Getting Started with Docker, Kubernetes and Google Kubernetes Engine
- Step 02 - Creating Google Cloud Account
- Step 03 - Creating Kubernetes Cluster with Google Kubernete Engine (GKE)
- Step 04 - Review Kubernetes Cluster and Learn Few Fun Facts about Kubernetes
- Step 05 - Deploy Your First Spring Boot Application to Kubernetes Cluster
- Step 06 - Quick Look at Kubernetes Concepts - Pods, Replica Sets and Deployment
- Step 07 - Understanding Pods in Kubernetes
- Step 08 - Understanding ReplicaSets in Kubernetes
- Step 09 - Understanding Deployment in Kubernetes
- Step 10 - Quick Review of Kubernetes Concepts - Pods, Replica Sets and Deployment
- Step 11 - Understanding Services in Kubernetes
- Step 12 - Quick Review of GKE on Google Cloud Console
- Step 13 - Understanding Kubernetes Architecture - Master Node and Nodes
- Step 14 - Understand Google Cloud Regions and Zones
- Step 15 - Installing GCloud
- Step 16 - Installing Kubectl
- Step 17 - Understand Kubernetes Rollouts
- Step 18 - Generate Kubernetes YAML Configuration for Deployment and Service
- Step 19 - Understand and Improve Kubernetes YAML Configuration
- Step 20 - Using Kubernetes YAML Configuration to Create Resources
- Step 21 - Understanding Kubernetes YAML Configuration - Labels and Selectors
- Step 22 - Quick Fix to reduce release downtime with minReadySeconds
- Step 23 - Understanding Replica Sets in Depth - Using Kubernetes YAML Config
- Step 24 - Configure Multiple Kubernetes Deployments with One Service
- Step 25 - Playing with Kubernetes Commands - Top Node and Pod
- Step 26 - Delete Hello World Deployments
- Step 27 - Quick Introduction to Microservices - CE and CC
- Step 28 - Deploy Microservices to Kubernetes
- Step 29 - Understand Environment Variables created by Kubernetes for Services
- Step 30 - Microservices and Kubernetes Service Discovery - Part 1
- Step 31 - Microservices and Kubernetes Service Discovery - Part 2 DNS
- Step 32 - Microservices Centralized Configuration with Kubernetes ConfigMaps
- Step 33 - Simplify Microservices with Kubernetes Ingress - Part 1
- Step 34 - Simplify Microservices with Kubernetes Ingress - Part 2
- Step 35 - Delete Kubernetes Clusters

Commands

```
docker run -p 8080:8080 in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE
kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080
kubectl scale deployment hello-world-rest-api --replicas=3
kubectl delete pod hello-world-rest-api-58ff5dd898-62l9d
kubectl autoscale deployment hello-world-rest-api --max=10 --cpu-percent=70
kubectl edit deployment hello-world-rest-api #minReadySeconds: 15
kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE
```

```
gcloud container clusters get-credentials in28minutes-cluster --zone us-
central1-a --project solid-course-258105
kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-
api:0.0.1.RELEASE
kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080
kubectl set image deployment hello-world-rest-api hello-world-rest-
api=DUMMY_IMAGE:TEST
kubectl get events --sort-by=.metadata.creationTimestamp
kubectl set image deployment hello-world-rest-api
hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE
kubectl get events --sort-by=.metadata.creationTimestamp
kubectl get componentstatuses
kubectl get pods --all-namespaces
kubectl get events
kubectl get pods
kubectl get replicaset
kubectl get deployment
kubectl get service
kubectl get pods -o wide
kubectl explain pods
kubectl get pods -o wide
kubectl describe pod hello-world-rest-api-58ff5dd898-9trh2
kubectl get replicasets
kubectl get replicaset
kubectl scale deployment hello-world-rest-api --replicas=3
kubectl get pods
kubectl get replicaset
kubectl get events
kubectl get events --sort-by=.metadata.creationTimestamp
kubectl get rs
kubectl get rs -o wide
kubectl set image deployment hello-world-rest-api hello-world-rest-
api=DUMMY_IMAGE:TEST
kubectl get rs -o wide
kubectl get pods
kubectl describe pod hello-world-rest-api-85995ddd5c-msjsm
kubectl get events --sort-by=.metadata.creationTimestamp
kubectl set image deployment hello-world-rest-api
hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE
kubectl get events --sort-by=.metadata.creationTimestamp
kubectl get pods -o wide
kubectl delete pod hello-world-rest-api-67c79fd44f-n6c7l
kubectl get pods -o wide
kubectl delete pod hello-world-rest-api-67c79fd44f-8bhdt
kubectl get componentstatuses
kubectl get pods --all-namespaces
gcloud auth login
kubectl version
gcloud container clusters get-credentials in28minutes-cluster --zone us-
central1-a --project solid-course-258105
kubectl rollout history deployment hello-world-rest-api
kubectl set image deployment hello-world-rest-api
hello-world-rest-api=in28min/hello-world-rest-api:0.0.3.RELEASE --record=true
kubectl rollout undo deployment hello-world-rest-api --to-revision=1
kubectl logs hello-world-rest-api-58ff5dd898-6ctr2
kubectl logs -f hello-world-rest-api-58ff5dd898-6ctr2
kubectl get deployment hello-world-rest-api -o yaml
kubectl get deployment hello-world-rest-api -o yaml > deployment.yaml
kubectl get service hello-world-rest-api -o yaml > service.yaml
kubectl apply -f deployment.yaml
kubectl get all -o wide
kubectl delete all -l app=hello-world-rest-api
kubectl get svc --watch
kubectl diff -f deployment.yaml
```

```
kubectl delete deployment hello-world-rest-api
kubectl get all -o wide
kubectl delete replicaset.apps/hello-world-rest-api-797dd4b5dc
kubectl get pods --all-namespaces
kubectl get pods --all-namespaces -l app=hello-world-rest-api
kubectl get services --all-namespaces
kubectl get services --all-namespaces --sort-by=.spec.type
kubectl get services --all-namespaces --sort-by=.metadata.name
kubectl cluster-info
kubectl cluster-info dump
kubectl top node
kubectl top pod
kubectl get services
kubectl get svc
kubectl get ev
kubectl get rs
kubectl get ns
kubectl get nodes
kubectl get no
kubectl get pods
kubectl get po
kubectl delete all -l app=hello-world-rest-api
kubectl get all
kubectl apply -f deployment.yaml
kubectl apply -f ../currency-conversion/deployment.yaml
```

Terraform

Projects
- Provision EC2 based HTTP Servers with Load Balancer
- Provision AWS and Azure Kubernetes Clusters (Azure DevOps Pipelines)

Steps
- Step 01 - Creating and Initializing First Terraform Project
- Step 02 - Create AWS IAM User Access Key and Secret
- Step 03 - Configure Terraform Environment Variables for AWS Access Keys
- Step 04 - Creating AWS S3 Buckets with Terraform
- Step 05 - Playing with Terraform State - Desired, Known and Actual
- Step 06 - Playing with Terraform Console
- Step 07 - Creating AWS IAM User with Terraform
- Step 08 - Updating AWS IAM User Name with Terraform
- Step 09 - Understanding Terraform tfstate files in depth
- Step 10 - gitignore Terraform tfstate files
- Step 11 - Refactoring Terraform files - Variables, Main and Outputs
- Step 12 - Creating Terraform Project for Multiple IAM Users
- Step 13 - Playing with Terraform Commands - fmt, show and console
- Step 14 - Recovering from Errors with Terraform
- Step 15 - Understanding Variables in Terraform
- Step 16 - Creating Terraform Project for Understanding List and Map
- Step 17 - Adding Elements - Problem with Terraform Lists
- Step 18 - Creating Terraform Project for Learning Terraform Maps
- Step 19 - Quick Review of Terraform FAQ
- Step 20 - Understanding Creation of EC2 Instances in AWS Console
- Step 21 - Creating New Terraform Project for AWS EC2 Instances
- Step 22 - Creating New EC2 Key Pair and Setting Up
- Step 23 - Adding AWS EC2 Configuration to Terraform Configuration
- Step 24 - Installing Http Server on EC2 with Terraform - Part 1
- Step 25 - Installing Http Server on EC2 with Terraform - Part 2
- Step 26 - Remove hardcoding of Default VPC with AWS Default VPC
- Step 27 - Remove hardcoding of subnets with Data Providers
- Step 28 - Remove hardcoding of AMI with Data Providers
- Step 29 - Playing with Terraform Graph and Destroy EC2 Instances
- Step 30 - Creating New Terraform Project for AWS EC2 with Load Balancers
- Step 31 - Create Security Group and Classic Load Balancer in Terraform
- Step 32 - Review and Destroy AWS EC2 with Load Balancers
- Step 33 - Creating Terraform Project for Storing Remote State in S3
- Step 34 - Create Remote Backend Project for Creating S3 Buckets
- Step 35 - Update User Project to use AWS S3 Remote Backend
- Step 36 - Creating multiple environments using Terraform Workspaces
- Step 37 - Creating multiple environments using Terraform Modules

Commands Executed

```
brew install terraform
terraform --version
terraform version
terraform init
export AWS_ACCESS_KEY_ID=*******
export AWS_SECRET_ACCESS_KEY=*********
terraform plan
terraform console
terraform apply -refresh=false
terraform plan -out iam.tfplan
terraform apply "iam.tfplan"
terraform apply -target="aws_iam_user.my_iam_user"
```

```
terraform destroy
terraform validate
terraform fmt
terraform show
export TF_VAR_iam_user_name_prefix = FROM_ENV_VARIABLE_IAM_PREFIX
export TF_VAR_iam_user_name_prefix=FROM_ENV_VARIABLE_IAM_PREFIX
terraform plan -refresh=false -
var="iam_user_name_prefix=VALUE_FROM_COMMAND_LINE"
terraform apply -target=aws_default_vpc.default
terraform apply -target=data.aws_subnet_ids.default_subnets
terraform apply -target=data.aws_ami_ids.aws_linux_2_latest_ids
terraform apply -target=data.aws_ami.aws_linux_2_latest
terraform workspace show
terraform workspace new prod-env
terraform workspace select default
terraform workspace list
terraform workspace select prod-env
```

Ansible

Step By Step Details
- Step 01 - Creating EC2 Instances for Ansible - Manually and with Terraform
- Step 02 - Setting Ansible Project with cfg and ansible hosts
- Step 03 - Playing with Ansible Commands
- Step 04 - Playing with Ansible Host File and Custom Groups
- Step 05 - Creating an Ansible Playbook for Ping
- Step 06 - Understanding Ansible Terminology - Control Node, Managed Nodes, Inventory
- Step 07 - Creating New Ansible Playbook for Executing Shell Commands
- Step 08 - Playing with Ansible Variables
- Step 09 - Creating New Ansible Playbook for Understanding Ansible Facts
- Step 10 - Creating New Ansible Playbook for Installing Apache and Serving HTML
- Step 11 - Reuse and Executing Multiple Ansible Playbooks
- Step 12 - Understanding Conditionals and Loops with Ansible
- Step 13 - Configuring EC2 Dynamic Inventory with Ansible
- Step 14 - Creating AWS EC2 Instances with Ansible
- Step 15 - Providing Declarative Configuration with Ansible
- Step 16 - Deleting all AWS EC2 Instances

Prerequisites
- You can use which ever approach you are comfortable with
      - 1 Manually
      - 2 using Terraform
      - 3 EC2 Instances

- EC2 Keys - `ls ~/aws/aws_keys/default-ec2.pem`

- AWS CLI - `aws configure`
```sh
# or
export AWS_ACCESS_KEY_ID=**************
export AWS_SECRET_ACCESS_KEY=**************
```

- boto3 and botocore - For EC2 Dynamic Inventory and Creating EC2 Instances
```sh
# TEst
python
# boto3 quick start
> import boto3
> client = boto3.client('ec2')
```

Create EC2 Instances using Terraform

```
cd terraform/backup/09-multiple-ec2-instances
export AWS_ACCESS_KEY_ID=**************
export AWS_SECRET_ACCESS_KEY=**************
terraform init
terraform apply
ls ~/aws/aws_keys/ # Make sure that the keys file is present
```

Ansible Commands

```
cd /in28Minutes/git/devops-master-class/ansible
ansible --version
ansible -m ping all
```

```
ansible all -a "whoami"
ansible all -a "uname"
ssh -vvv -i ~/aws/aws_keys/default-ec2.pem ec2-user@3.83.104.44
ls ~/aws/aws_keys/default-ec2.pem
chmod 400 /Users/rangaraokaranam/aws/aws_keys/default-ec2.pem
ansible all -a "uname"
ansible all -a "uname -a"
ansible all -a "pwd"
ansible all -a "python --version"
ansible dev -a "python --version"
ansible qa -a "python --version"
ansible first -a "python --version"
ansible groupofgroups -a "python --version"
ansible devsubset -a "python --version"
ansible --list-host all
ansible --list-host dev
ansible --list-host first
ansible --list-host \!first
ansible --list-host qa:dev
ansible-playbook playbooks/01-ping.yml
ansible-playbook playbooks/02-shell.yml
ansible-playbook playbooks/03-variables.yml
ansible-playbook playbooks/03-variables.yml -e variable1=CommandLineValue
ansible-playbook playbooks/04-ansible-facts.yml
ansible-playbook playbooks/05-install-apache.yml
ansible-playbook playbooks/06-playbooks.yml
ansible-playbook playbooks/06-playbooks.yml --list-tasks
ansible-playbook playbooks/06-playbooks.yml --list-hosts
ansible-playbook playbooks/06-playbooks.yml --list-tags
ansible-playbook -l dev playbooks/01-ping.yml
ansible-playbook playbooks/07-conditionals-and-loops.yml
ansible-inventory --list
ansible-inventory --graph
ansible-playbook playbooks/08-dynamic-inventory-ping.yml
ansible-playbook playbooks/09-create-ec2.yml

```