<u>**Module Checklist**</u>

AWS Services

By Techworld with Nana

Video Overview

★ Introduction to Amazon Web Services
★ Create an AWS account
★ Identity & Access Management (IAM)
★ Regions & Availability Zones
★ Virtual Private Cloud (VPC)
★ Classless Inter-Domain Routing - CIDR Block explained
★ Introduction to Amazon Elastic Compute Cloud (EC2)
★ AWS and Jenkins Part I - Jenkins Pipeline to deploy on AWS EC2
★ AWS and Jenkins Part II - Deploy using Docker Compose (Docker-Compose, ECR)
★ AWS and Jenkins Part III - Complete Pipeline (Docker-Compose, ECR, Dynamic versioning)
★ Introduction to AWS CLI
★ AWS and Infrastructure as Code (Terraform)
★ Container Services Preview

Demo Projects	
Java Maven Project	https://gitlab.com/nanuchi/java-maven-app
Web App Project	https://github.com/nanuchi/react-nodejs-example

Check your progress… 1/9

Introduction to Amazon Web Services

❏ Watched video

Useful Links:

- AWS Services Overview: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/aws-overview.pdf

Create an AWS account

❏ Watched video

❏ **Demo executed** - create an AWS Free Tier account

Useful Links:

- Step by Step instruction on how to create and activate a new Amazon Web Services account? https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/
- Services included in Free Tier: https://aws.amazon.com/free/?all-free-tier&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc

Pricing of Services we use:

- EC2 Pricing: https://aws.amazon.com/ec2/pricing/
- S3 Pricing: https://aws.amazon.com/s3/pricing/
- VPC Pricing: https://aws.amazon.com/vpc/pricing/
- ECR Pricing: https://aws.amazon.com/ecr/pricing/
- IAM Pricing: https://aws.amazon.com/iam/faqs/#Pricing (always free)
- EKS Pricing: https://aws.amazon.com/eks/pricing/

Check your progress... 2/9

Identity and Access Management - IAM explained

- ❏ Watched video
- ❏ **Demo executed** - Created Admin IAM User

Best Practice:

- Assign the permission (policy) to the Role, rather than on the User directly
- Give User the **least privilege** they need

Regions and Availability Zones

- ❏ Watched video

Useful Links:

- Amazon's Regions & Availability Zones: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

Virtual Private Cloud - VPC explained

- ❏ Watched video

Classless Inter-Domain Routing - CIDR explained

- ❏ Watched video

Useful Links:

- IP Calculator: http://jodies.de/ipcalc?host=10.0.0.0&mask1=1&mask2=
- Calculate sub-cidr blocks: http://www.davidc.net/sites/default/subnets/subnets.html

Check your progress… 3/9

Introduction to Amazon Elastic Compute Cloud (EC2)

- ❑ Watched video
- ❑ **Demo executed - Deploy WebApp Container manually on EC2 Instance:**
    - ❑ Stored Private Key in .ssh folder
    - ❑ EC2 Instance created
    - ❑ Built and pushed Docker Image to your private DockerHub
    - ❑ Docker installed and run Docker Image
    - ❑ Security Group configured: Opened Firewall to access Web App from Browser

Useful Links:

- Webapp project - to be built and pushed to your private DockerHub repo: https://github.com/nanuchi/react-nodejs-example

Check your progress... 4/9

AWS & Jenkins Part I - Jenkins Pipeline to deploy on AWS EC2

- ❏ **Watched video**
- ❏ **Demo 1 executed - Deploy WebApp Container via Jenkins Pipeline on EC2 Instance:**
  - ❏ Installed SSH agent plugin on Jenkins
  - ❏ Created ssh credentials type for EC2 on Jenkins
  - ❏ Configured Jenkinsfile to use the sshAgent and execute docker run command on EC2
  - ❏ Docker Login to DockerHub or your other private Docker Repository (if you haven't already)
  - ❏ Security Group configured: Added Jenkins IP Address and opened port to access Web App from Browser
  - ❏ Deploy Webapp on EC2 Instance by executed Multi-Branch Pipeline
  - ❏ Access Application on port 3080 in the browser
- ❏ **Demo 2 executed - Deploy Java Maven App via Jenkins Pipeline on EC2 Instance:**
  - ❏ Configured Jenkinsfile to build and deploy on EC2 Instance
  - ❏ Executed Multi-Branch Pipeline on Jenkins
- ❏ **Bonus Learning:**
  - ❏ Try to deploy an App with docker-compose and run a shell script

Useful Links:

- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/blob/feature/jenkinsfile-sshagent

Check your progress... 5/9

AWS & Jenkins Part II - Deploy using Docker Compose (Docker-Compose, ECR)

- ❏ **Watched video**
- ❏ **Demo executed - Deploy Java Maven App via Jenkins Pipeline on EC2 Instance using Docker-Compose File:**
    - ❏ Installed Docker-Compose on EC2 Instance
    - ❏ Created docker-compose.yaml file
    - ❏ Configured Jenkinsfile to execute docker-compose command
    - ❏ Executed Jenkins Pipeline and deploy to AWS EC2 Instance
    - ❏ Improvement: Extract to Shell Script

Useful Links:

- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/blob/feature/jenkinsfile-sshagent
- Docker-Compose Download (AWS and Jenkins Part II): https://docs.docker.com/compose/install/

AWS & Jenkins Part III - Complete Pipeline (Docker-Compose, ECR, Dynamic versioning)

- ❏ **Watched video**
- ❏ **Demo executed - as before with dynamic versioning:**
    - ❏ Adjusted Jenkinsfile to include dynamic versioning
    - ❏ Executed Jenkins Pipeline and deploy to AWS EC2 Instance

Useful Links:

- Java-maven-app - sshagent: https://gitlab.com/nanuchi/java-maven-app/-/blob/feature/jenkinsfile-sshagent
- Java-maven-app - version increment: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs/Jenkinsfile-version-increment

Check your progress... 6/9

Introduction to AWS CLI

- ❏ Watched video
- ❏ **Demo executed - install and configure AWS CLI**
- ❏ **Demo executed - using EC2 commands:**
  - ❏ Created Security Group
  - ❏ Created SSH key pair
  - ❏ Created EC2 Instance
  - ❏ SSHed into newly created EC2 Instance
  - ❏ Used filter and query options
- ❏ **Demo executed - using IAM commands:**
  - ❏ Created User
  - ❏ Created Group
  - ❏ Added User to Group
  - ❏ Assigned policy to Group
  - ❏ Created credentials for new User
  - ❏ Created a new Policy and assigned to newly created Group
  - ❏ Logged in with new User in AWS UI
  - ❏ Created access keys for newly created User
- ❏ **Demo executed - delete AWS resources created before**

Useful Links:

- AWS CLI Installation Instructions for different OS: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
- AWS CLI User Guide: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-using.html
- Listing and Filtering your resources: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Filtering.html
- AWS Shell - The interactive productivity booster for the AWS CLI: https://github.com/awslabs/aws-shell

Check your progress... 7/9

Useful commands:

- EC2 Service Commands

```
## List all available security-group ids
aws ec2 describe-security-groups

## create new security group
aws ec2 describe-vpcs
aws ec2 create-security-group --group-name my-sg --description "My security group" --vpc-id
vpc-1a2b3c4d

## this will give output of created my-sg with its id, so we can do:
aws ec2 describe-security-groups --group-ids sg-903004f8

## add firewall rule to the group for port 22
aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 22 --cidr
203.0.113.0/24
aws ec2 describe-security-groups --group-ids sg-903004f8


# Use an existing key-value pair or if you want, create and use a new key-pair. 'KeyMaterial' gives
us an unencrypted PEM encoded RSA private key.
aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text > MyKeyPair.pem

# launch ec2 instance in the specified subnet of a VPC
aws ec2 describe-subnets
aws ec2 describe-instances -> will give us ami-imageid, we will use the same one
aws ec2 run-instances
    --image-id ami-xxxxxxxx
    --count 1
    --instance-type t2.micro
    --key-name MyKeyPair
    --security-group-ids sg-903004f8
    --subnet-id subnet-6e7f829e


# ssh into the ec2 instance with the new key pem after creating it - public IP will be returned as
json, so query it
aws ec2 describe-instances --instance-ids {instance-id}
chmod 400 MyKeyPair.pem
ssh -i MyKeyPair.pem ec2-user@public-ip

# check UI for all the components that got created

# describe-instances - with filter and query
--filter is for picking some instances. --query is for picking certain info about those instances
```

Check your progress... 8/9

Useful commands:

- IAM Service Commands

```
# same way as ec2 had a bunch of commands for components relevant for ec2 instances, iam does too
aws iam create-group --group-name MyIamGroup
aws iam create-user --user-name MyUser
aws iam add-user-to-group --user-name MyUser --group-name MyIamGroup

# verify that my-group contains the my-user
aws iam get-group --group-name MyIamGroup

# attach policy to group
## this is the command so we need the policy-ARN - how can we get that?
aws iam attach-user-policy --user-name MyUser --policy-arn {policy-arn} - attach to user directly
aws iam attach-group-policy --group-name MyGroup --policy-arn {policy-arn} - attach policy to group

## let's go and check on UI AmazonEC2FullAccess policy ARN

## OR if you know the name of the policy 'AmazonEC2FullAccess', list them
aws iam list-policies --query 'Policies[?PolicyName==`AmazonEC2FullAccess`].{ARN:Arn}' --output text
aws iam attach-group-policy --group-name MyGroup --policy-arn {policy-arn}

# validate policy attached to group or user
aws iam list-attached-group-policies --group-name MyGroup - [aws iam list-attached-user-policies
--user-name MyUser]

# Now that user needs access to the command line and UI, but we didn't give it any credentials. So
let's do that as well!
## UI access
aws iam create-login-profile --user-name MyUser --password My!User1Login8P@ssword
--password-reset-required
-> user will have to update password on UI or programmatically with command: aws iam
update-login-profile --user-name MyUser --password My!User1ADifferentP@ssword

# Create test policy
aws iam create-policy --policy-name bla --policy-document file://bla.json

## cli access
aws iam create-access-key --user-name MyUser
-> you will see the access keys
```

Check your progress… 9/9

Useful commands:

```
## Now let's ssh into the EC2 instance with this user
'aws configure' with new user creds

$ aws configure set aws_access_key_id default_access_key
$ aws configure set aws_secret_access_key default_secret_key

export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_DEFAULT_REGION=us-west-2

## Now let's login with this user on UI and see what got created!

### NOTES at the end
Revert to admin user creds
Delete all the stuff
```

AWS & Infrastructure as Code (Terraform) Preview

❏ Watched video

Container Services Preview

❏ Watched video