Please prepare a presentation to discuss these points. You don’t need to implement these.

* What type of steps you would perform in order to verify the deployment is successful?

We can implement these steps in our CICD pipeline. For example the following bash code from the buildspec file:

      - TD\_REVISION=$(aws ecs describe-services --cluster my-ecs-cluster --service Microservice\_A \

        --query "services[].taskDefinition" | jq -r ".[0]")

      - echo Current running task definition revision - $TD\_REVISION

This will display the last revision of our task definition for this microservice.

After the update of our microservice we can repeat the same commands to check if the microservice is running with newer revision of the same task definition.

We can connect to API endpoint to check the version of our service using the following commands:

      - echo Check micorservice version

      - VERSION=$(curl -s http://load\_balancer\_FQDN:PORT/API\_ENDPOINT) #this is for demo purpose

      - echo Microservice\_A running version - $VERSION

* Plan and a task break-down how you would implement monitoring of this deployed app

We can you CloudWatch to look after the desired count of the instances of our microservice, if we have less running instances of this service running CloudWatch can trigger SNS to inform us that there is an issue with this service.

We can create bash script using aws-cli or python script using boto3 (AWS SDK for python). This script can check for the same thing (count of running instances, CPU load, memory, connections (for this we need to check the loadbalancer infront our service), after that this script can do some parsing of these statistics and we can pass them to monitoring system.

* What kind of security policies and scans would you recommend to put into place?

For security we can use IAM roles and policies in order to restrict our running services. This way we can grand them access only to resources and services that we want.

We can also use Sonarqube for static code analysis and linters on merge requests. This can be done using webhook that will trigger CICD pipeline – if the pipeline is ok we can merge the code, otherwise the code need to be fixed before merging.

* What other improvements would you make to the CI/CD process if you had more time?

Implement security scans in the current example. Some sort of templating based on environment (one values for dev, another for prod).

Disable monitoring of the service before update. Enable monitoring after update.

* Are there any other good dev-ops tools and practices that you recommend?

We can use terraform to deploy our resources no matter of the cloud provider. Also we can replace the bash commands in the current pipeline with ansible playbooks and roles.

Implement some notification channel between our CICD tool and our messenger (teams, slack, etc.).