CLOUD COMPUTING APPLICATIONS

Docker on Amazon Elastic Container Service (ECS)

Prof. Reza Farivar

Amazon Elastic Container Service (ECS)

- Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service
- Containers either run on customer-managed EC2 instances, or on AWS Fargate
- Fargate: Serverless backend for container deployment
  - AWS manages resource provisioning for container instances
- Services such as Amazon Sagemaker and Lex internally run on ECS

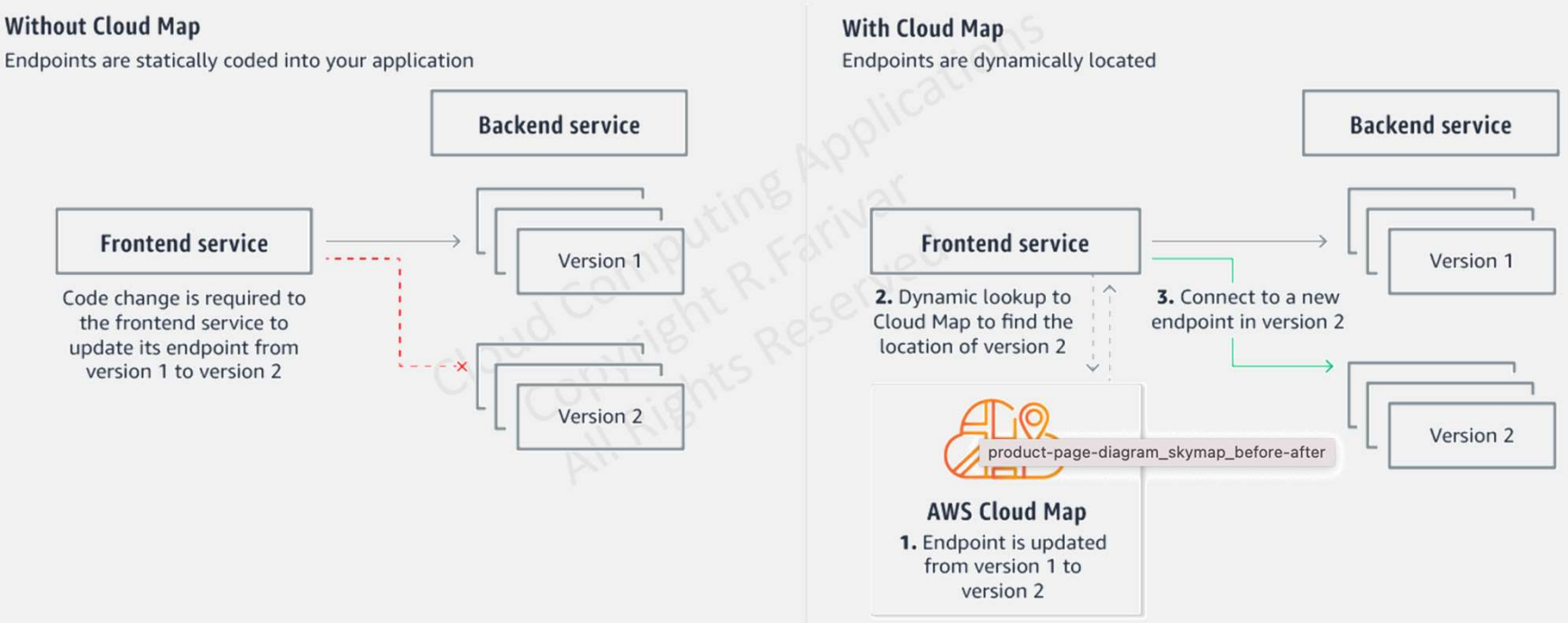
ECS Task Definition

- The Docker image to use with each container in your task
- How much CPU and memory to use with each task or each container within a task
- The launch type to use, which determines the infrastructure on which your tasks are hosted
- The Docker networking mode to use for the containers in your task
- The logging configuration to use for your tasks
- Whether the task should continue to run if the container finishes or fails
- The command the container should run when it is started
- Any data volumes that should be used with the containers in the task
- The IAM role that your tasks should use

Example AWS ECS Task Definition

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Ar
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-de
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "requiresCompatibilities": [
      "FARGATE"
   ]
}
```

Service Discover: AWS Cloud Map



Using Docker Compose File Syntax in ECS

- The `ecs-cli compose` and `ecs-cli compose service` commands allow you to create task definitions and manage your Amazon ECS tasks and services using Docker Compose files
- Additional ECS parameters specified for the container size parameters in a separate YAML file

Example

```
version: '3'
services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    logging:
      driver: awslogs
      options:
        awslogs-group: /ecs/cli/tutorial
        awslogs-region: us-east-1
        awslogs-stream-prefix: nginx
```

hello-world.yml

```
version: 1
task_definition:
  services:
    nginx:
      cpu_shares: 256
      mem_limit: 0.5GB
      mem_reservation: 0.5GB
```

ecs-params.yml

```
ecs-cli compose --project-name hello-world
--file hello-world.yml --ecs-params ecs-params.yml
--region us-east-1 create --launch-type EC2
```

Docker AWS ECS context

- Another method: Directly use docker
  - Set up an AWS context in one Docker command
    - `$ docker context create ecs myecscontext`
  - Use Compose files
    - `--context myecscontext`
    - `docker compose up --context myecscontext`
- Docker translates the compose application to CloudFormation template
  - Actual mapping:
    - https://github.com/docker/compose-cli/blob/main/docs/ecs-architecture.md

Native AWS Services

- ECS cluster for the Compose application
- Use the default VPC
  - A Security Group per network in the Compose file on the AWS account's default VPC
  - LoadBalancer to route traffic to the services
- Service Discovery (service-to-service load balancing) is handled by AWS Cloud Map
- Volumes are handled by instantiating EFS filesystems
- Secrets handled by AWS Secrets Manager

```yaml
services:
  webapp:
    image: ...
    secrets:
      - foo

secrets:
  foo:
    name: "arn:aws:secretsmanager:eu-west-3:1234:secret:foo-ABC123"
    external: true
```

Docker integration with Azure

- Docker is also integrated with Azure Container Instances
  - `$ docker login azure`
  - `$ docker context create aci myacicontext`
  - `$ docker --context myacicontext run -p 80:80 nginx`
- Volumes → Azure File Share
- Port mapping → Only symmetrical mapping 80:80
- Networks not supported
  - At least not until the recording of this video in 2021
  - Communication between services is implemented by defining mapping for each service in the shared `/etc/hosts` file of the container group.