upGrad
#LifeKoKaroLift

AWS ECS Advanced

A Production-Ready System in AWS

Course: Containerisation at Scale

Lecture On: AWS ECS Advanced

Instructor: Dipesh Garg

upGrad

Prerequisite for this Session

Basic understanding of docker

Basic understanding of container orchestration

Basic understanding of AWS ECS

Basic understanding of AWS cloud components: VPC, EC2, S3, IAM

Today’s Agenda

upGrad

- Recap ECS
- Environment variables
- ECS IAM roles
- Task placement
- Service updates in ECS - Rolling updates
- ECS load balancing and autoscaling
- ECS production-ready setup demo:
  - Custom VPC, public and private subnets, IGW, NAT and route tables
  - EC2 mode ECS cluster
  - Task definition and services
  - Monitoring, logging, constraints and service discovery
- Learn the KPIs for DevOps engineers

upGrad

AWS ECS - Quick Recap

AWS ECS

The AWS Elastic Container Service (AWS ECS) is a container management service that can quickly launch, manage and exit docker containers on a cluster.

This means ECS helps us manage:

- The phases in the life cycle of a container, which include:
    - The starting phase,
    - The running phase and
    - The deletion phase.

AWS Architecture

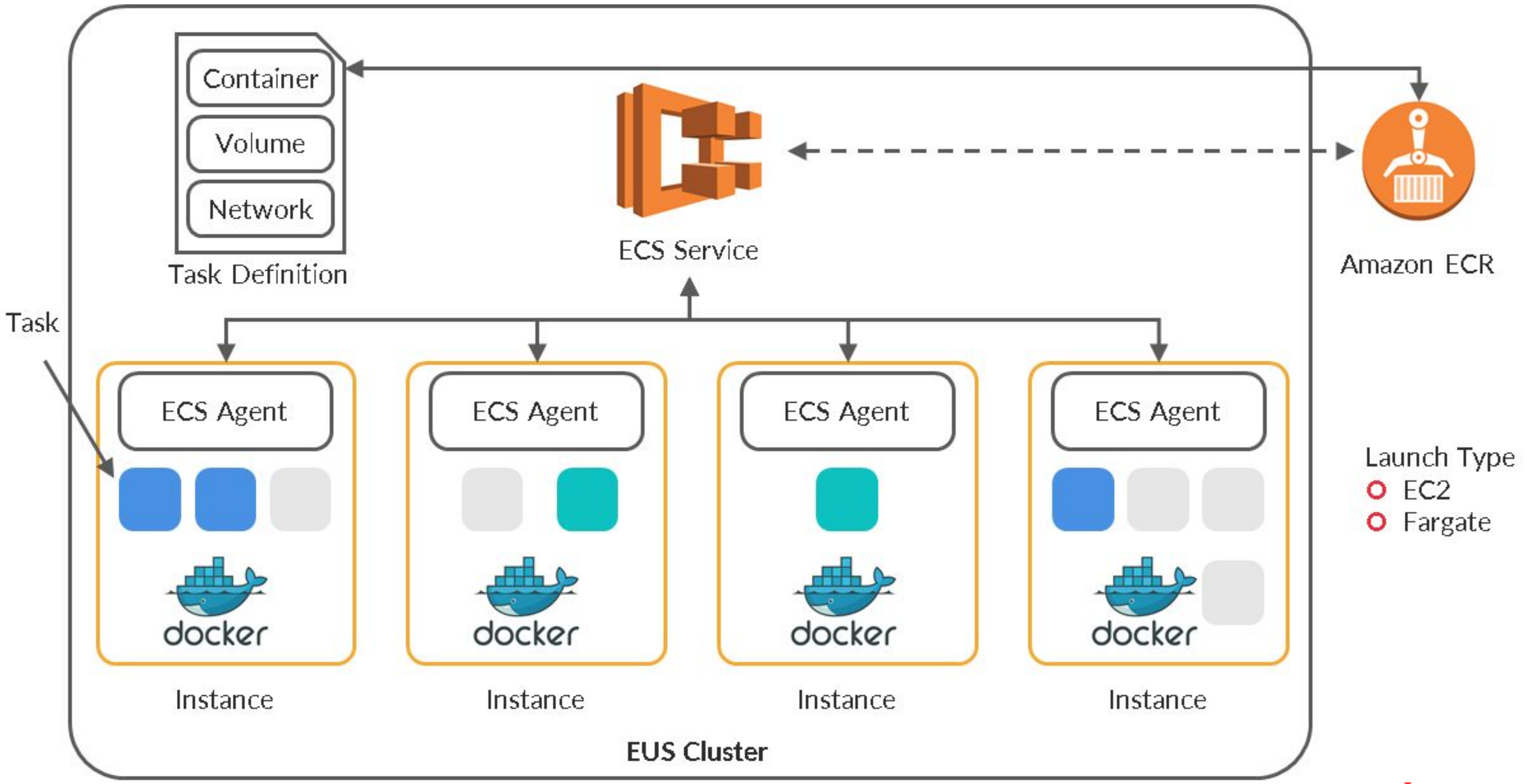
- ECS cluster
  - A set of EC2 instances - One or more EC2 instances
  - EC2 mode/Fargate mode

- Task definition
  - Contains multiple sub-components/sub-definitions for running a container
  - Docker image name, container name, port, logging, etc.

- Services
  - Enables you to run and maintain a specified number of task definitions
  - Has two modes: Replica and daemon

ECS Architecture

upGrad



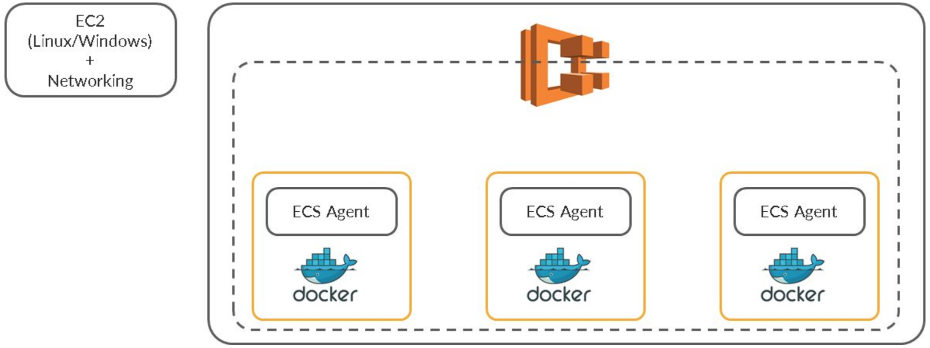
Launch Type

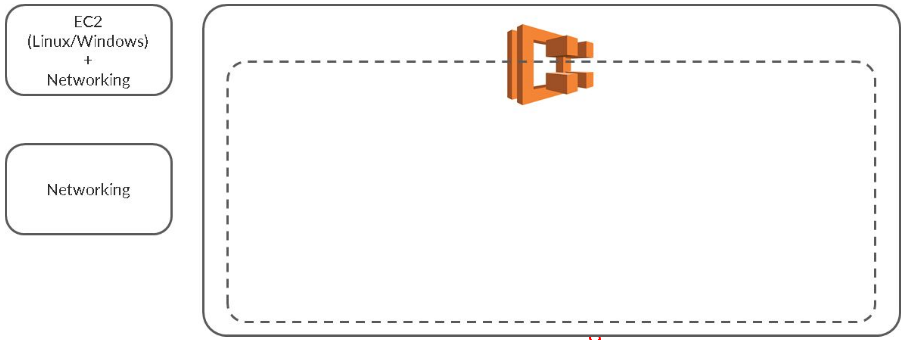
- EC2
- Fargate

ECS Cluster Launch Types

upGrad

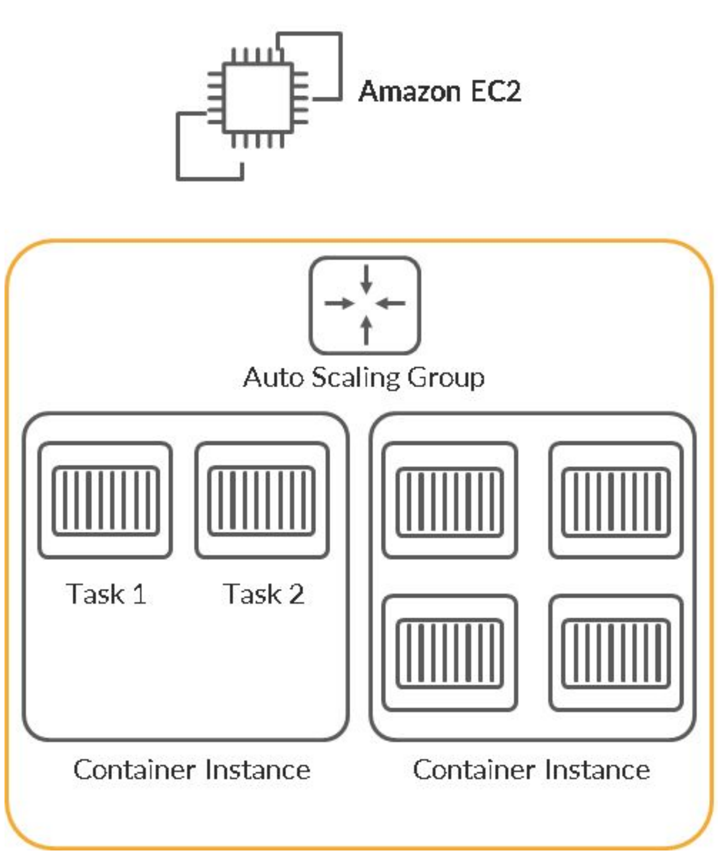
- It determines the type of infrastructure on which your services and tasks are placed:
    - EC2 launch type
    - Fargate launch type

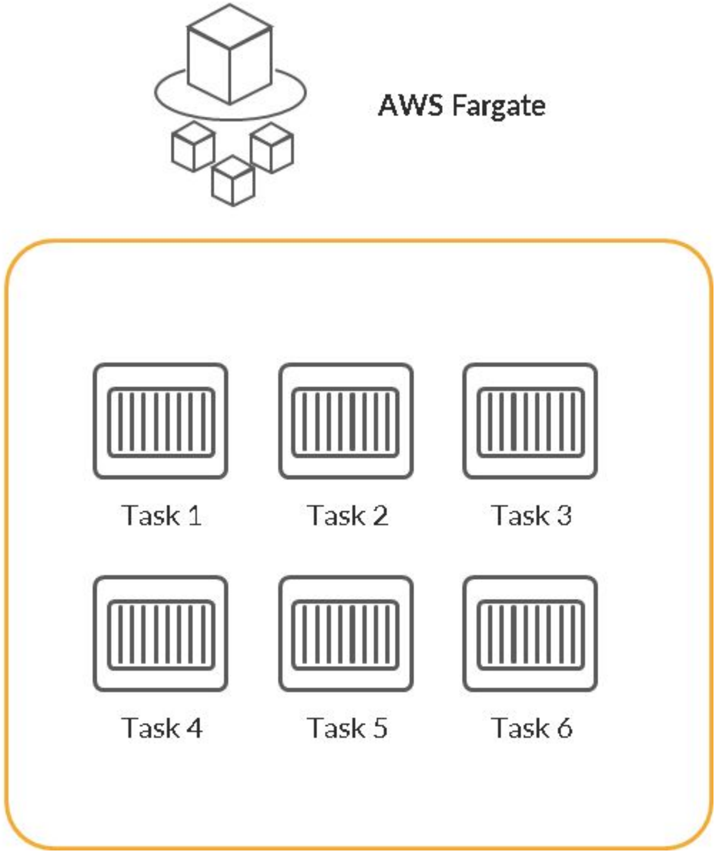


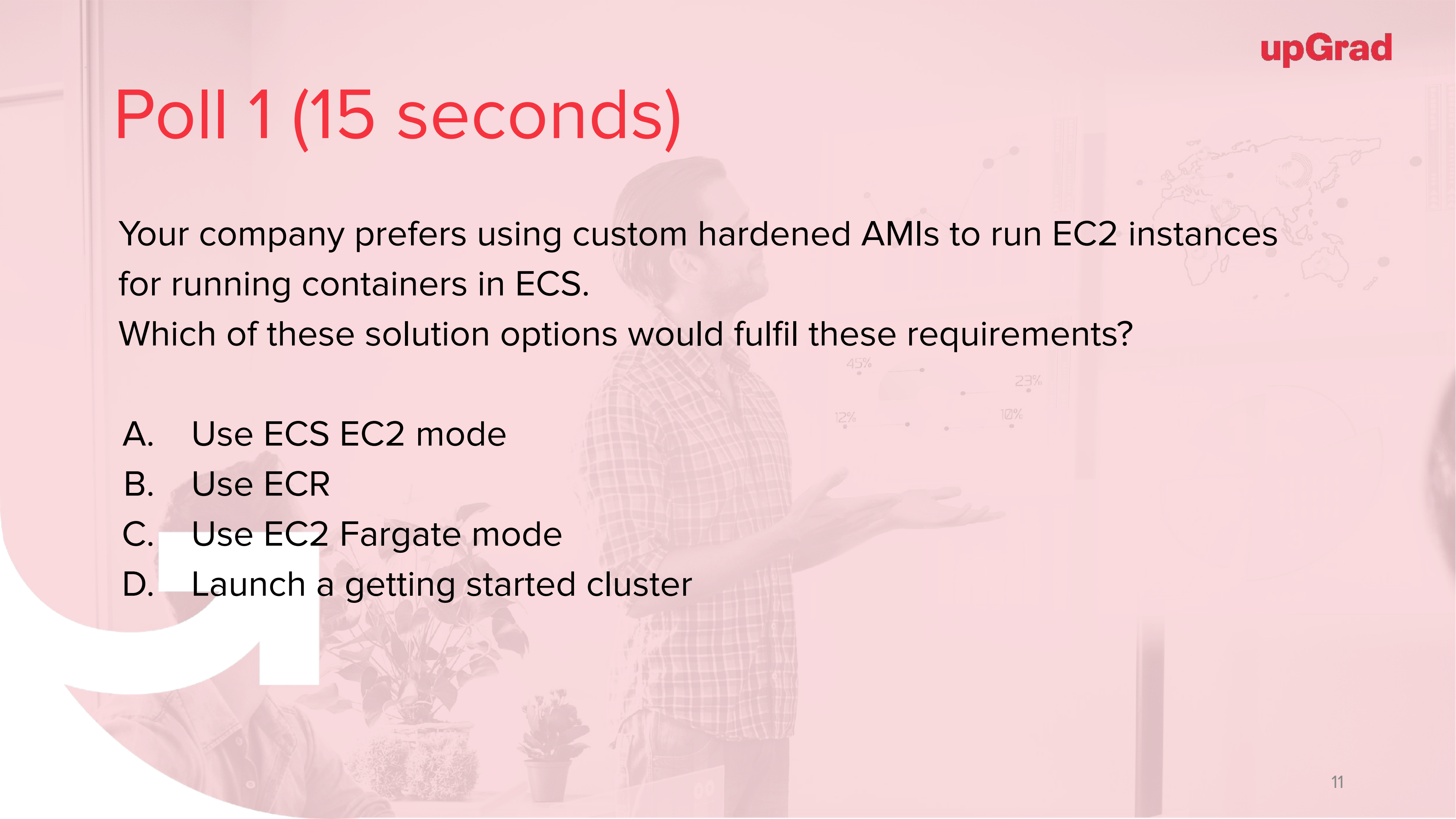


ECS Cluster Launch Types

upGrad



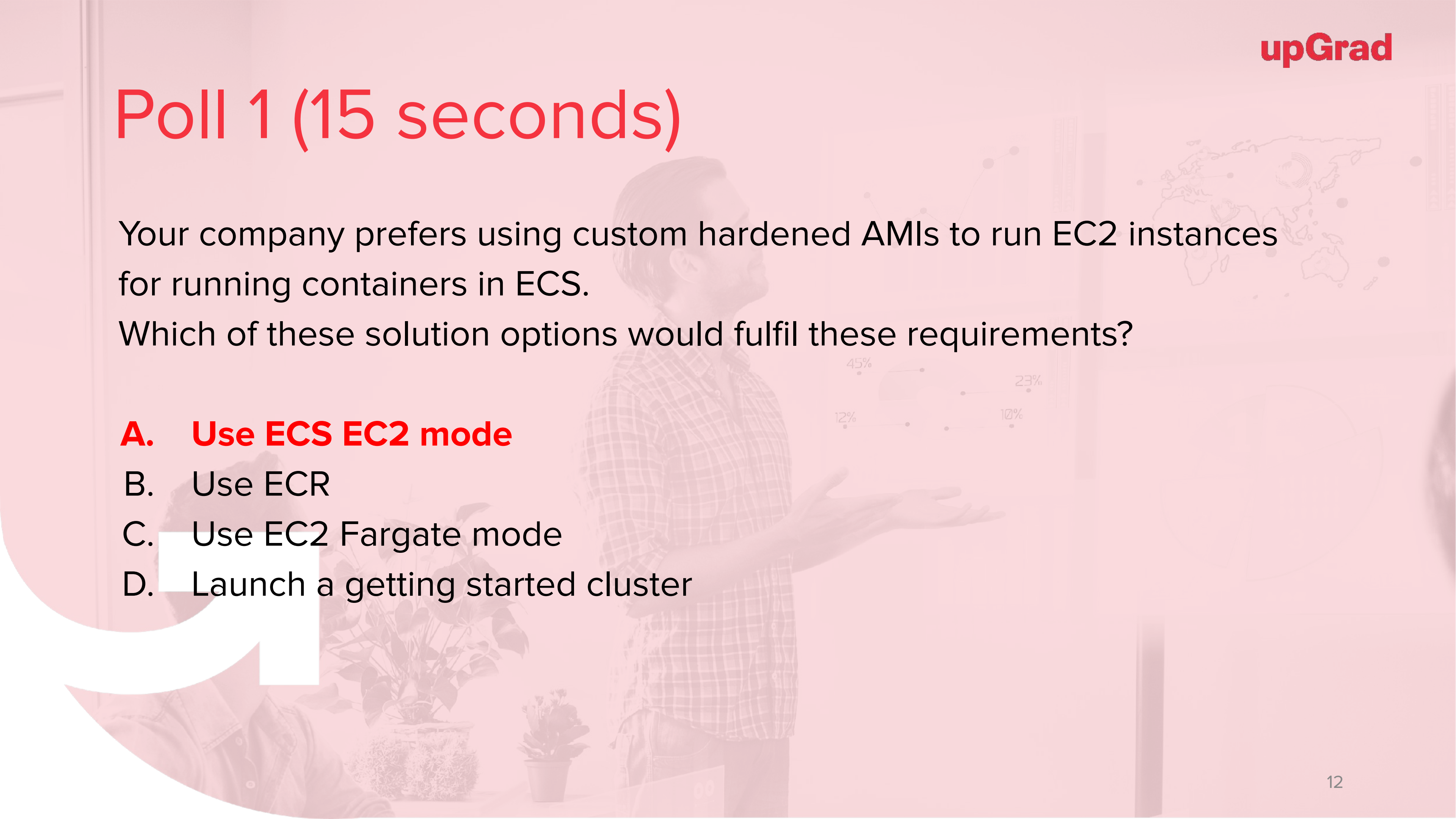




Poll 1 (15 seconds)

Your company prefers using custom hardened AMIs to run EC2 instances for running containers in ECS.
Which of these solution options would fulfil these requirements?

A. Use ECS EC2 mode
B. Use ECR
C. Use EC2 Fargate mode
D. Launch a getting started cluster



Poll 1 (15 seconds)

Your company prefers using custom hardened AMIs to run EC2 instances for running containers in ECS.
Which of these solution options would fulfil these requirements?

A. Use ECS EC2 mode
B. Use ECR
C. Use EC2 Fargate mode
D. Launch a getting started cluster

Poll 2 (15 seconds)

A company is not sure about the pricing to use the Elastic Container Service (ECS) with the EC2 launch type. Which of these statements regarding the pricing for this service is correct?

A. It is based on the number of master + data nodes along with the EBS cost.

B. It is based on the EC2 instances and the EBS volumes used.

C. It is based on the CPU + memory container used per hour.

D. It is based on the master + EC2 usage.

upGrad

Poll 2 (15 seconds)

A company is not sure about the pricing to use the Elastic Container Service (ECS) with the EC2 launch type. Which of these statements regarding the pricing for this service is correct?

A. It is based on the number of master + data nodes along with the EBS cost.

B. It is based on the EC2 instances and the EBS volumes used.

C. It is based on the CPU + memory container used per hour.

D. It is based on the master + EC2 usage.

AWS ECS Advanced: ECS Tasks

ECS Task Components

- Standard: Container name, image
- Memory limits (soft and hard)
- Port mappings
- Network settings
- Storage
- Logging
- Labels
- Environment variables
- Task IAM roles
- Placement constraints

Environment Variables

upGrad

- Variables whose values are defined outside the program
- Made up of key–value pairs and are part of the complete environment in which a particular process or application runs

- Benefits:
  - They make configuration management easy
  - **Improvement in security:** You can save confidential value in secret managers, instead of hard coding them in the applications.
  - A single point of management leads to fewer chances of mistakes.

Example: Environment variables such as db host:port, db username and password are passed while launching the task.

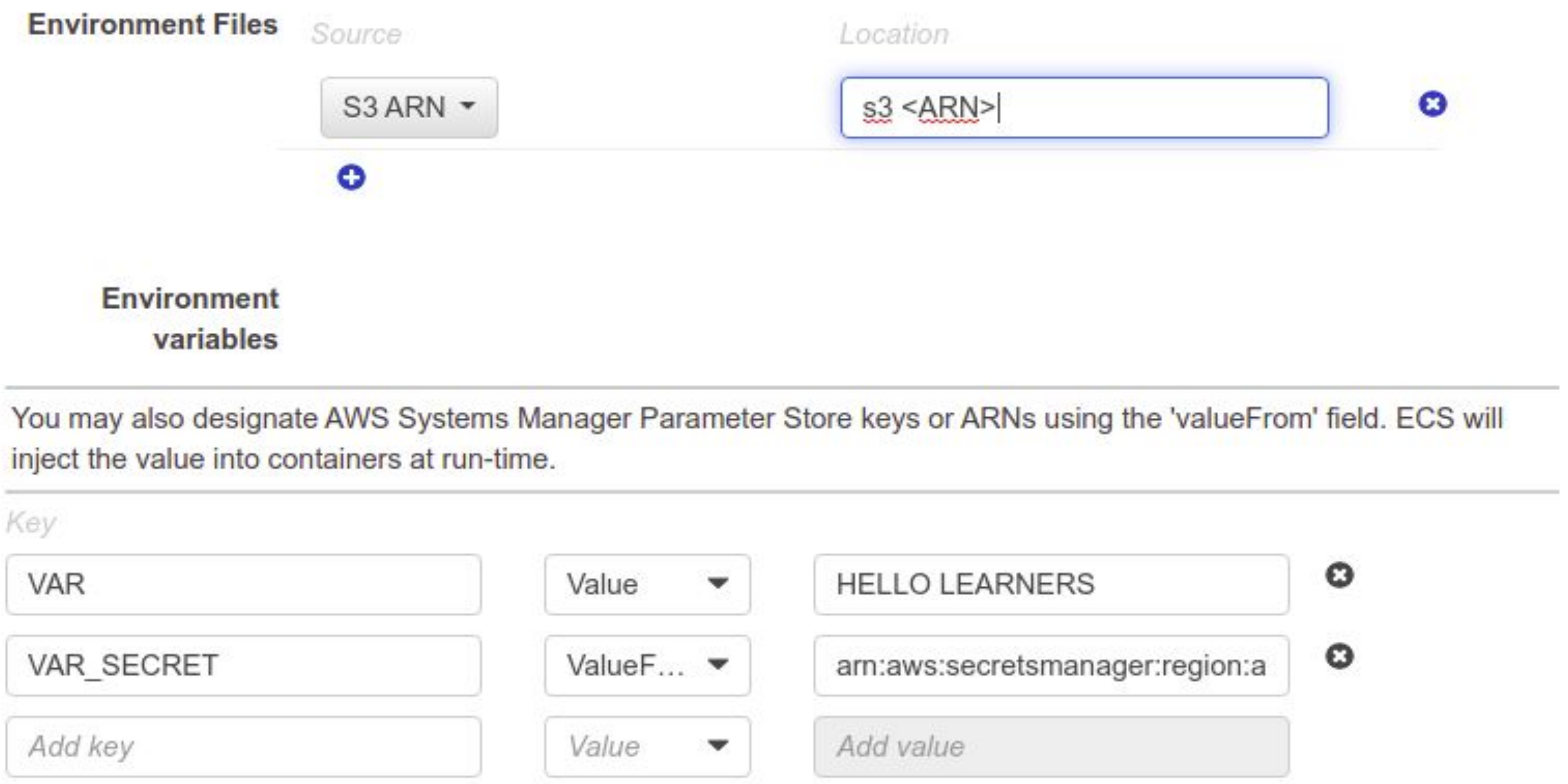
```
export VAR="Hello learners to session 3.2"
echo $VAR
```

Environment Variables in ECS

- You can inject environment variables in a container using:
    - ECS task definition
        - You need to push a new task definition for any change.
        - Environment variables are hard coded while creating the task definition; so, it is not ideal for secrets.
    - S3
        - Values get resolved at container launch.
        - It is helpful for bulk uploading of environment variables.
    - SSM/Secret Manager
        - Values get resolved at container launch.
        - It should be used for credentials.
        - It is easy to change values/offer autorotation.

Environment Variables in ECS

upGrad



Environment Files

Source	Location
S3 ARN	s3 <ARN>

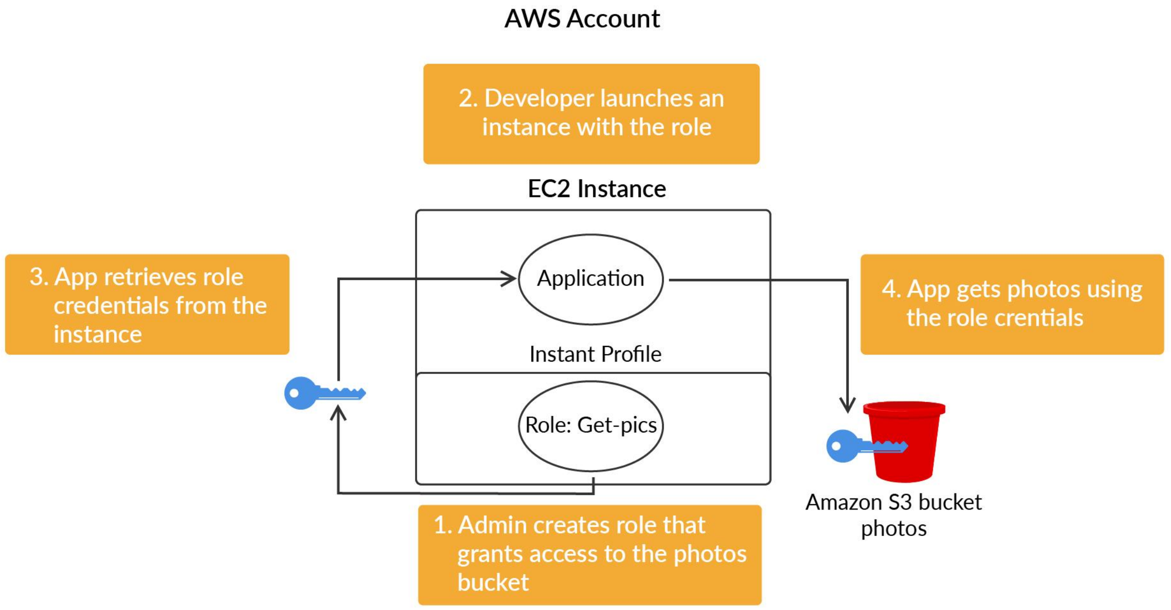
Environment variables

You may also designate AWS Systems Manager Parameter Store keys or ARNs using the 'valueFrom' field. ECS will inject the value into containers at run-time.

Key		
VAR	Value	HELLO LEARNERS
VAR_SECRET	ValueF…	arn:aws:secretsmanager:region:a
Add key	Value	Add value

IAM Roles

- AWS identities that have specific permissions
- **Example:** If your application wants to read/write some files from S3

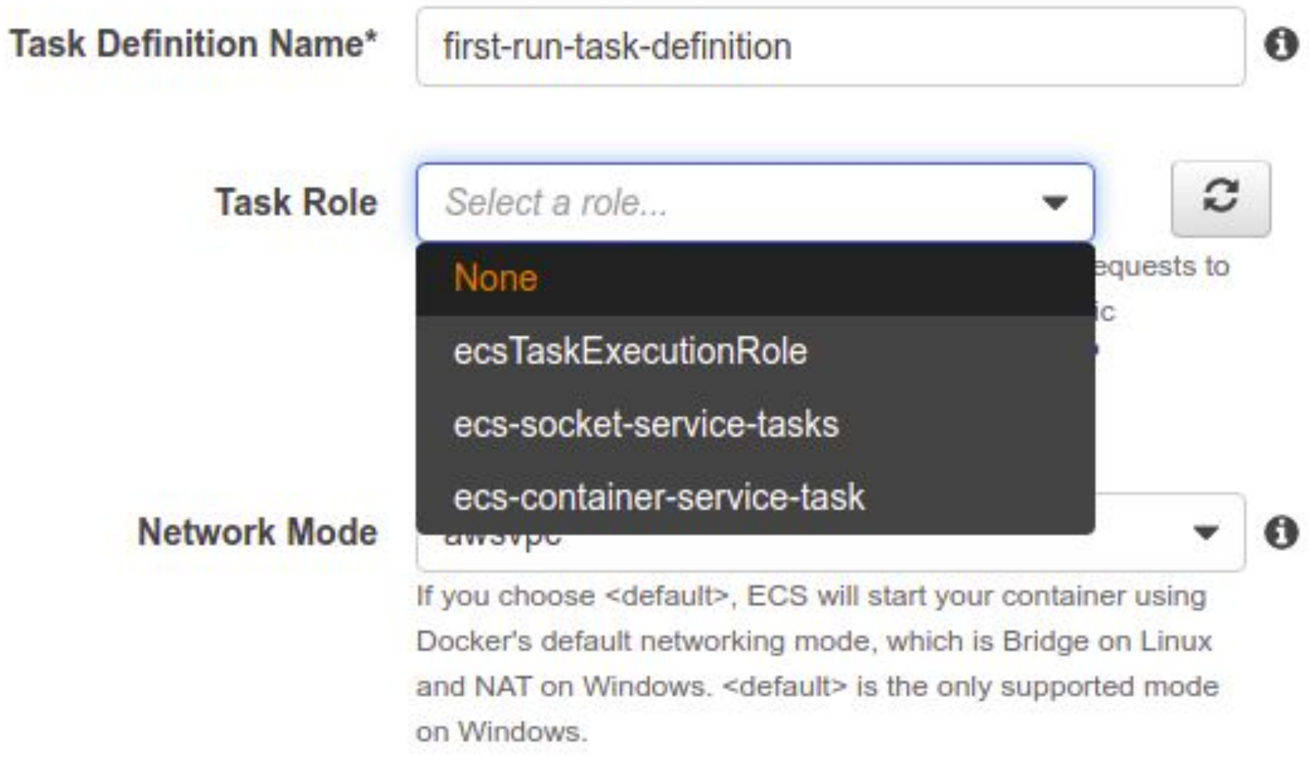

AWS Account
2. Developer launches an instance with the role
EC2 Instance
Application
Instant Profile
Role: Get-pics
3. App retrieves role credentials from the instance
4. App gets photos using the role crentials
Amazon S3 bucket photos
1. Admin creates role that grants access to the photos bucket

IAM Roles in ECS

- You can attach IAM roles in a container using:
    - The EC2 instance profile
        - Used by the ECS agent to make API calls to the ECS service
        - Automatically created for you in the console first-run experience
    - The ECS task role
        - Allow each task to have a separate role
        - Highly secure since the container of one task does not have access to the credentials intended for the container of another task

IAM Roles in ECS

upGrad



Task Definition Name* first-run-task-definition

Task Role Select a role...

None
ecsTaskExecutionRole
ecs-socket-service-tasks
ecs-container-service-task

equests to
ic

Network Mode awsvpc

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. <default> is the only supported mode on Windows.

ECS Task Placement

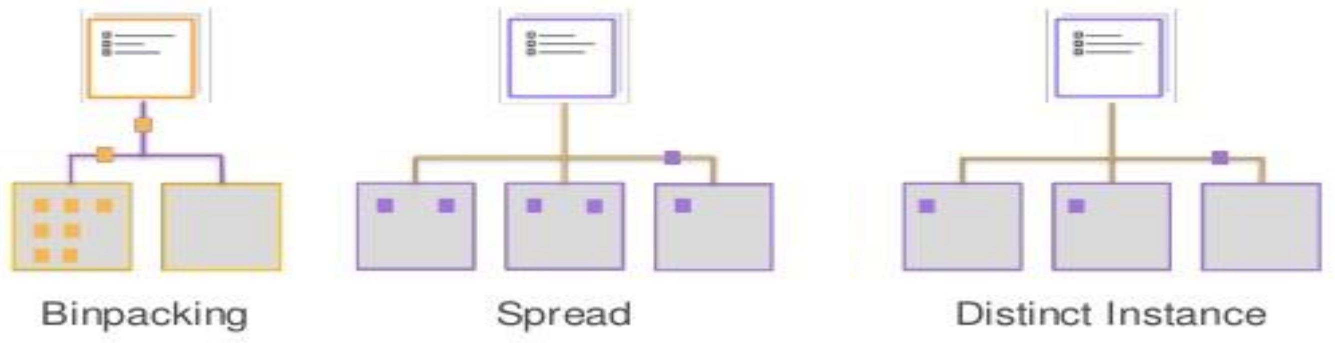
- An algorithm for selecting instances for:
  - Placement of new tasks
    - Imagine you have five instances in the ECS cluster, and you want to launch a new task. On which instance should this task be placed?
  - Termination of tasks
    - Imagine you are running three tasks on a five-node ECS cluster and want to kill one of these tasks. Which task should be terminated?

- Not supported for tasks using the Fargate launch type since EC2 nodes are not managed manually in the Fargate mode.
- Achieved by:
  - A placement strategy and
  - A placement constraint.

ECS Task Placement Strategy

upGrad

- Selecting instances for placement or termination of tasks
- Example:
  - Select instances such that tasks are distributed evenly across a group of instances

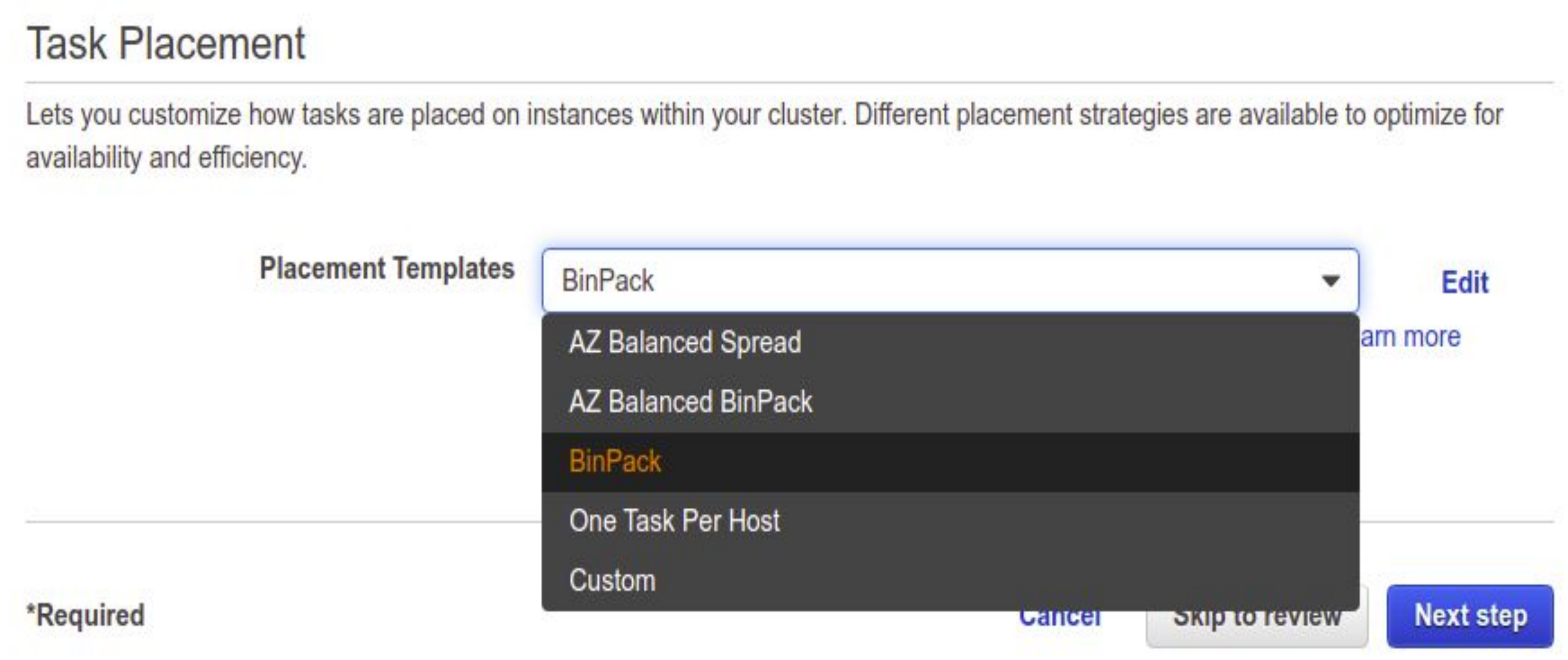


ECS Task Placement Strategy Types

- Binpacking
  - Tasks are placed in EC2 so as to leave the least amount of unused CPU or memory
- Spread
  - Tasks are placed evenly based on the specified value.
  - **Example:** Spread across three availability zones
- One task per host
  - It would place only one task per instance.
- Random
  - Tasks are placed randomly.

ECS Task Placement Strategy Types

upGrad



Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

Placement Templates: BinPack

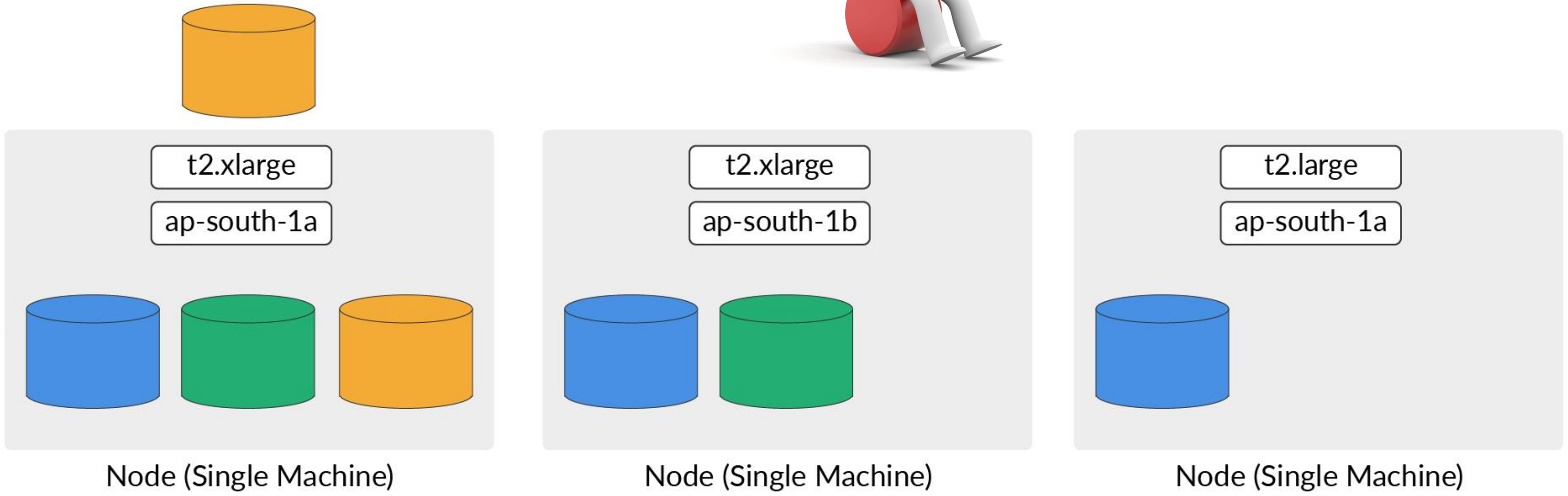
ECS Task Placement Constraint

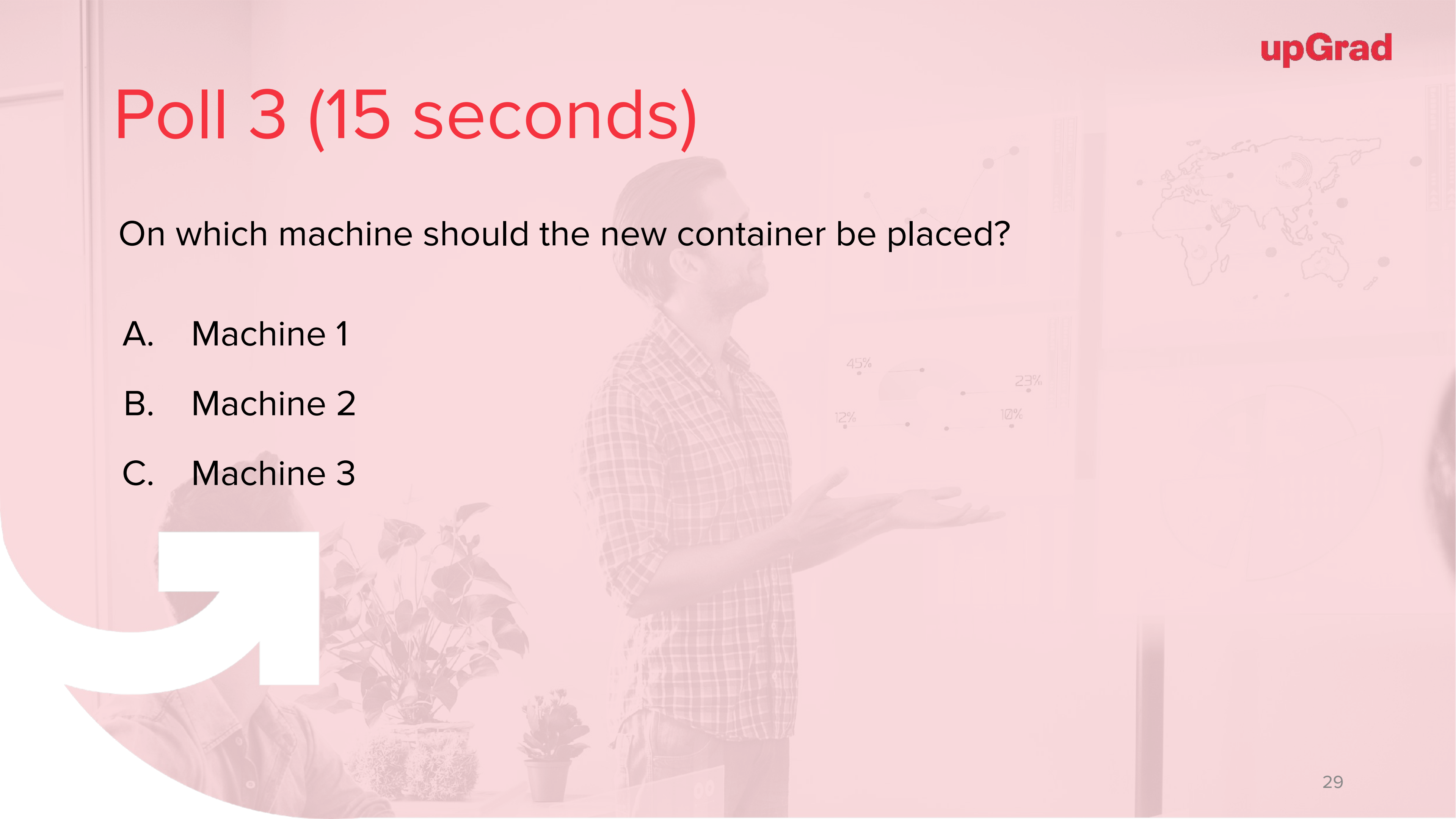
- It is a rule that is considered during task placement.
- It allows you to filter the EC2 instances used for the placement of tasks.
- ECS first filters the container instances that match the constraints and then applies the placement strategy to place the tasks.
- Examples:
  - Instance type should be m5.xlarge
  - Availability Zone should be ap-south-1a

ECS Task Placement

Where should the pending task get placed under t following configuration?

Placement strategy: AZ balanced spread
Placement constraint: Instance type should be t2.xl





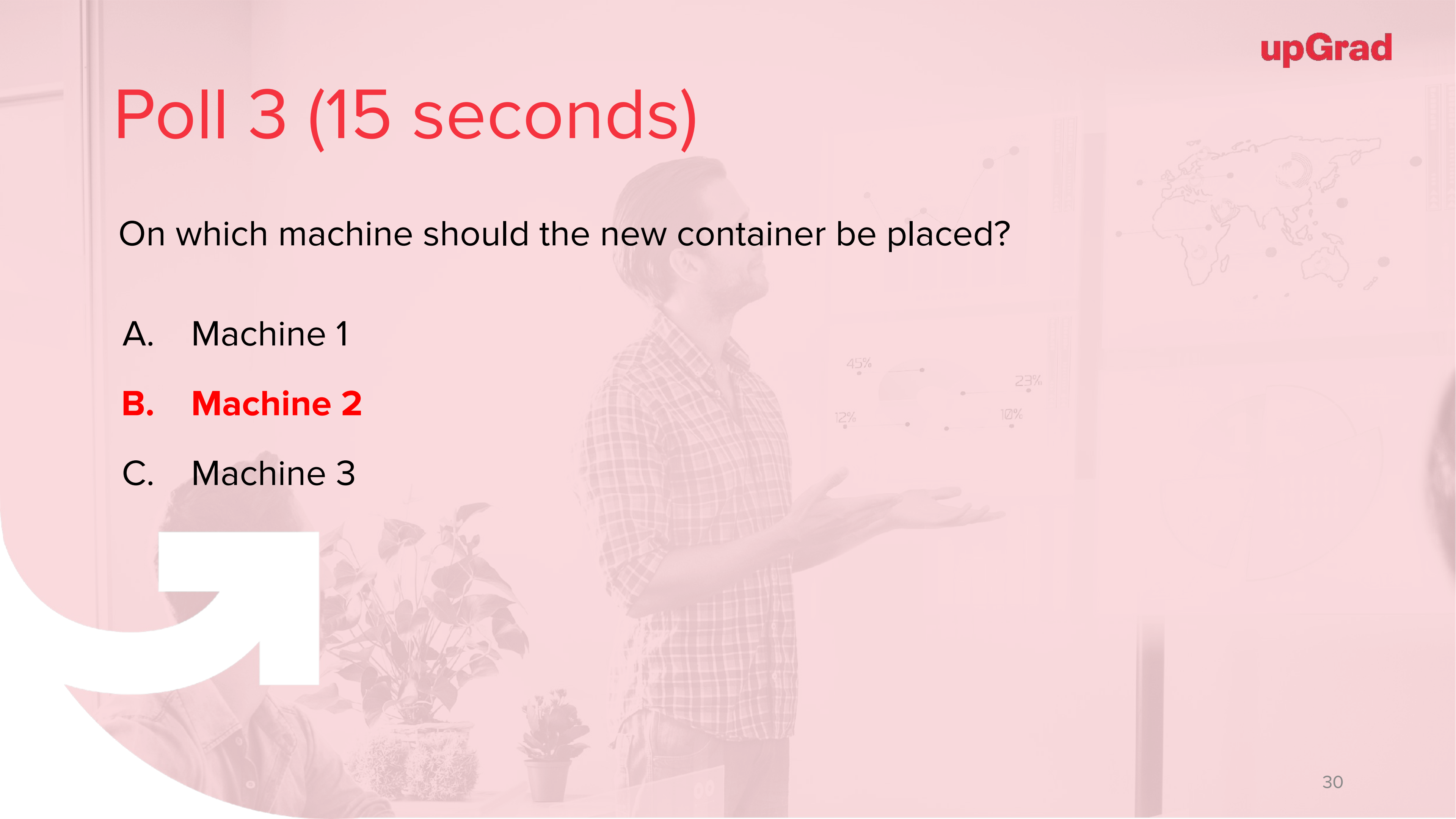
Poll 3 (15 seconds)

On which machine should the new container be placed?

A. Machine 1

B. Machine 2

C. Machine 3



Poll 3 (15 seconds)

On which machine should the new container be placed?

A. Machine 1

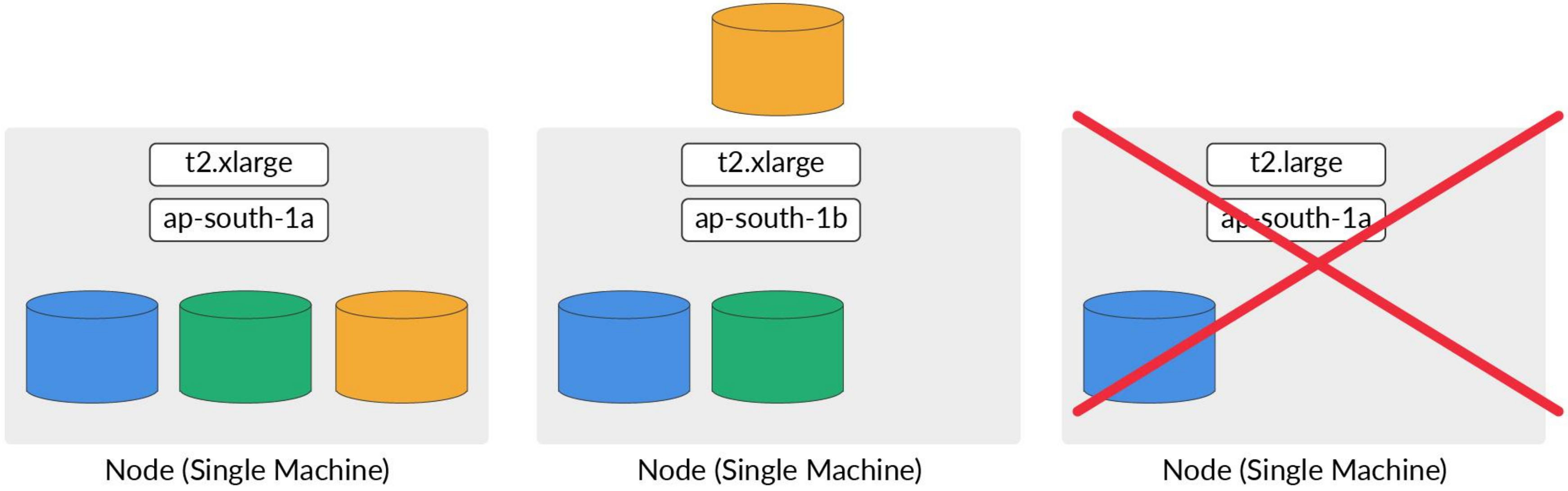
B. Machine 2

C. Machine 3

ECS Task Placement

Where should the pending task get placed under the following configuration?

Placement constraint: Instance type should be t2.xlarge.
Placement strategy: AZ balanced spread

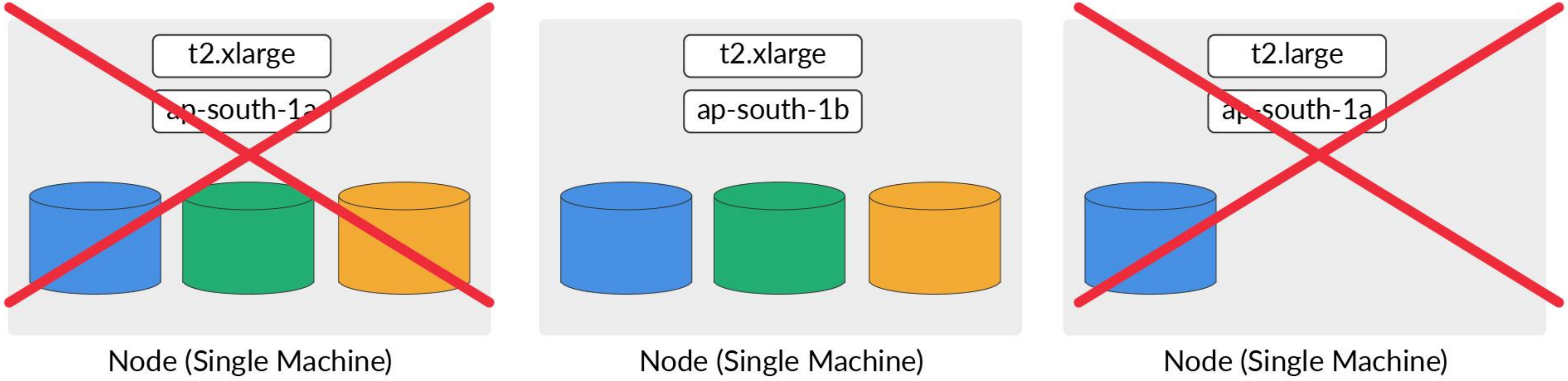


ECS Task Placement

Where should the pending task get placed under the following configuration?

Placement constraint: Instance type should be t2.xlarge.

Placement strategy: AZ balanced spread



Demo 1: ECS Task

- ECS task environment variables
- Task IAM roles
- Task placement strategies

AWS ECS Advanced: ECS Service

ECS Service Components

upGrad

- Standard: Service name, task version
- Type (replica anddaemon)
- Service discovery
- Deployment configuration
- Service autoscaling
- Load balancing

Deployment

upGrad

- What is deployment?
  - It refers to setting up an application for the first time.
  - It also refers to updating the application from an older version to a newer version.
  - It can be done to send new features/updates, bug fixes, configuration changes, etc.

- During deployment, consider the following points:
  - It should be a zero-downtime deployment. Users should not be impacted during the deployment process.
  - New versions should be tested in non-production environments before sending them to the production environment.
  - Parameters such as application logs, and the db's and application's CPU and memory usage should be monitored while deploying new features or bug fixes. After deployment, you should test whether everything is working fine

AWS ECS Deployment Types

- Deployment strategies supported in AWS ECS
  - **Rolling update**
    - The Amazon ECS service scheduler replaces currently running tasks with new tasks.
    - The number of tasks that Amazon ECS adds or removes from the service during a rolling update is controlled by the deployment configuration.
    - As traffic comes into the application, some users may interact with the new code, while others may land on an older version.
    - It works well for IT organisations with hosting infrastructure large enough for some servers to be taken out of service.
    - It is a cost-efficient method since you do not need additional infrastructure resources to deploy new changes.

AWS ECS Deployment Types

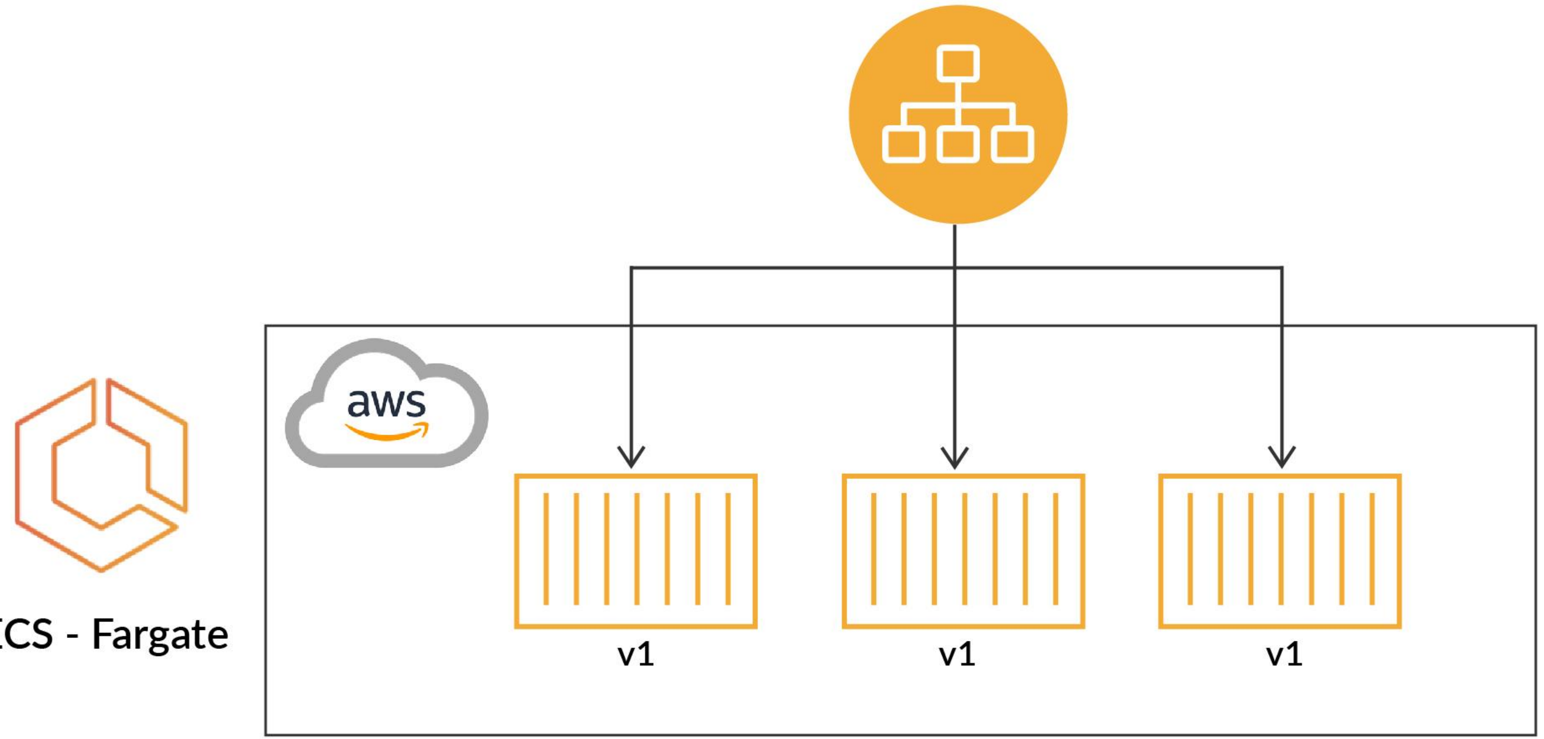
- Deployment strategies supported in AWS ECS
  - **Blue-Green**
    - It enables you to verify a new deployment of a service before sending production traffic to it.
    - This strategy requires IT organisations to maintain two identical hosting environments, each large enough for production. Infrastructure costs double.
    - It is useful for testing new features before sending the complete traffic to the new version.

AWS ECS Deployment Type: Rolling Updates

- The number of tasks that AWS ECS adds or removes from the service during a rolling update is controlled by the deployment configuration.
- Deployment configuration consists of:
    - minimumHealthyPercent
        - Minimum limit on the number of tasks that should be running
        - **Example:** If the minimum limit is 60% and the desired count is 10, then ECS will maintain at least six healthy container count and can stop four containers for deployment.
    - maximumPercent
        - Maximum limit on the number of tasks that should be running
        - **Example:** If the maximum limit is 150% and the desired count is 10, then ECS can launch five new containers before stopping the old containers, thus making a total count of 15.

AWS ECS Deployment Type: Rolling Updates

upGrad

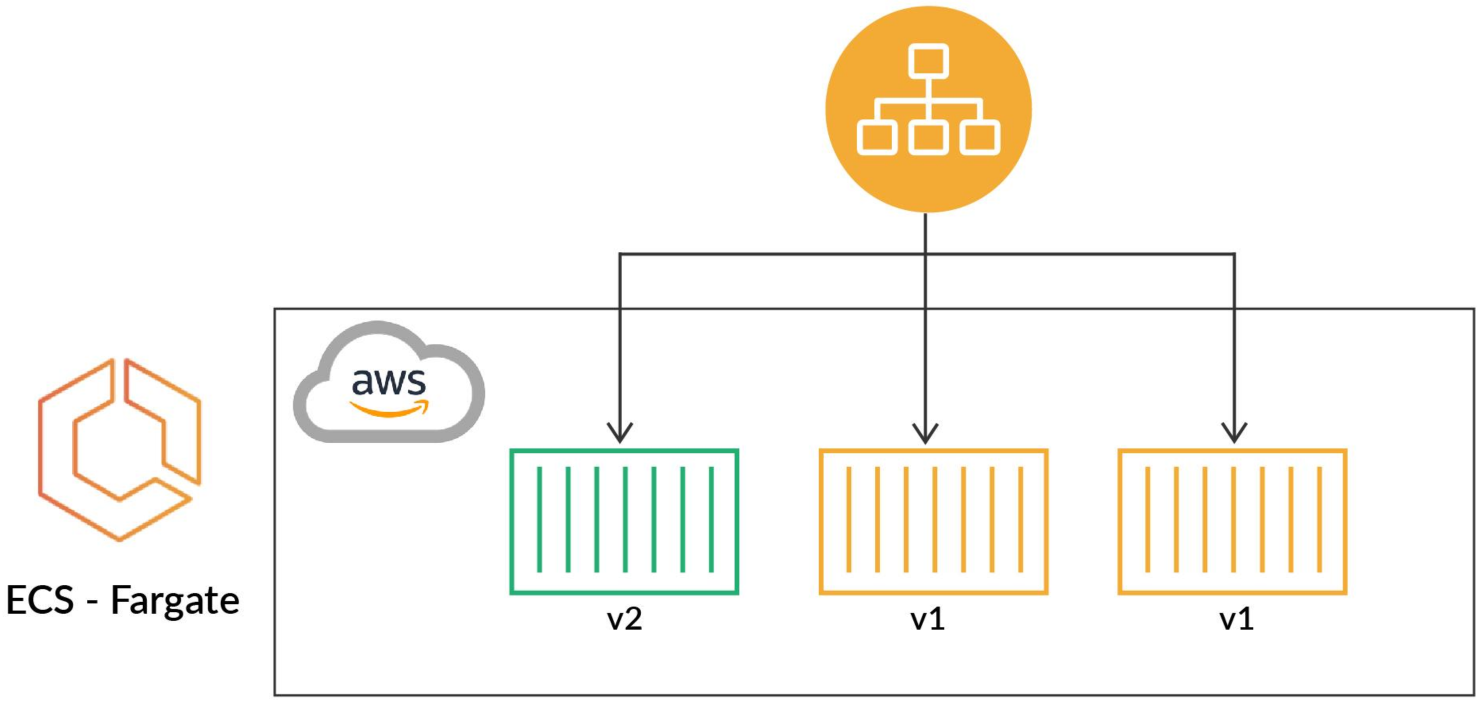


AWS ECS Deployment Type: Rolling Updates

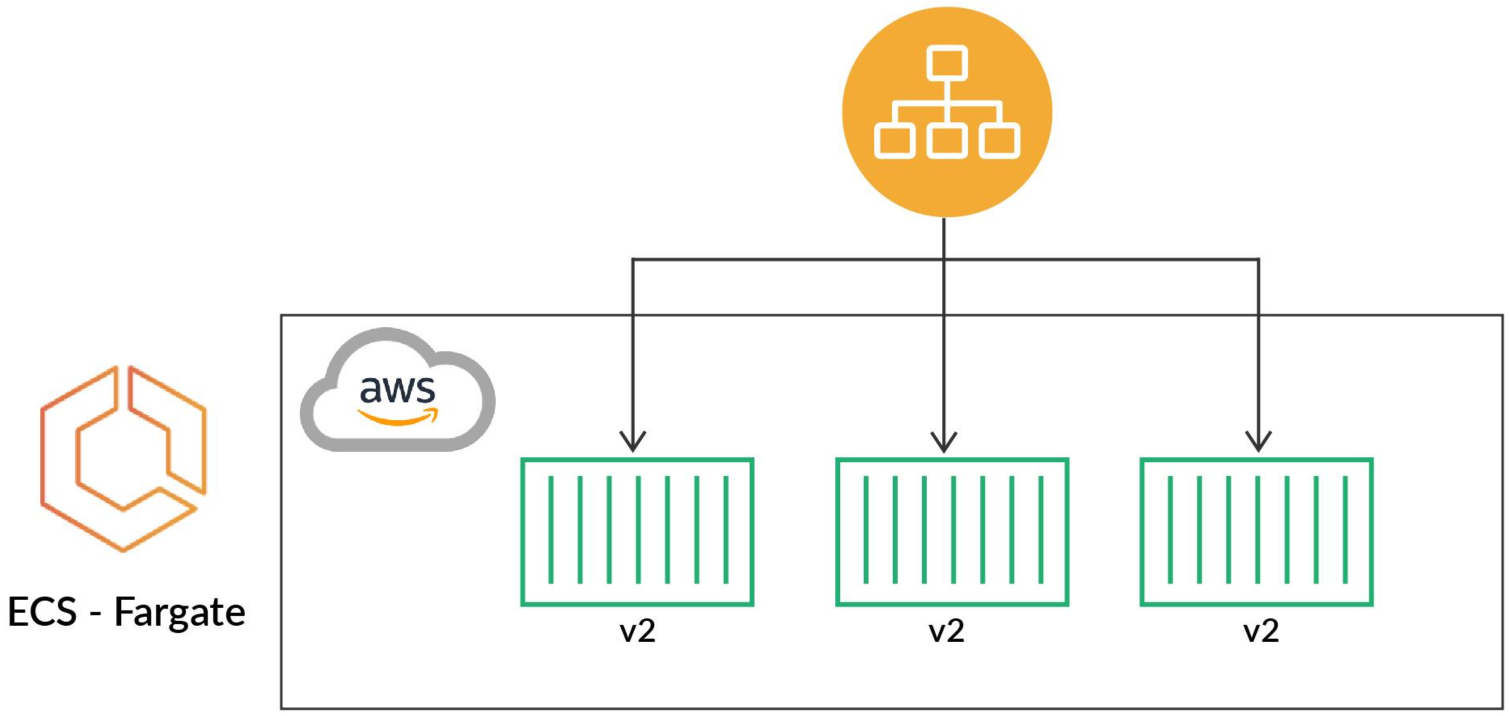
- **Deployment strategy:** Rolling updates

- minimumHealthyPercent: 60
- maximumHealthyPercent: 100

AWS ECS Deployment Type: Rolling Updates



AWS ECS Deployment Type: Rolling Updates



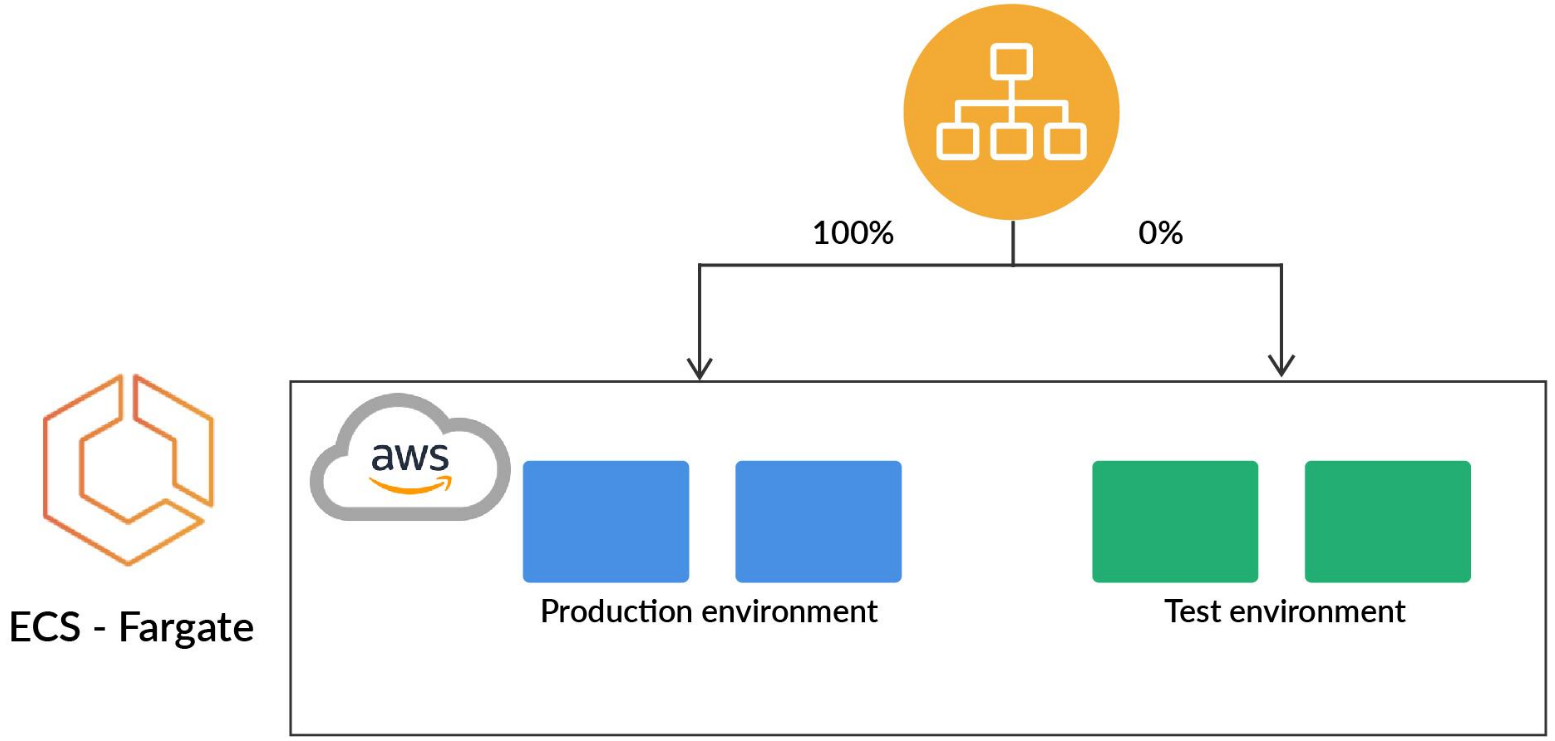
AWS ECS Deployment Type: Blue-Green

- It enables you to verify a new deployment of a service before sending production traffic to it.

- Two parallel versions of an application run: Older - **blue** and newer - **green**.

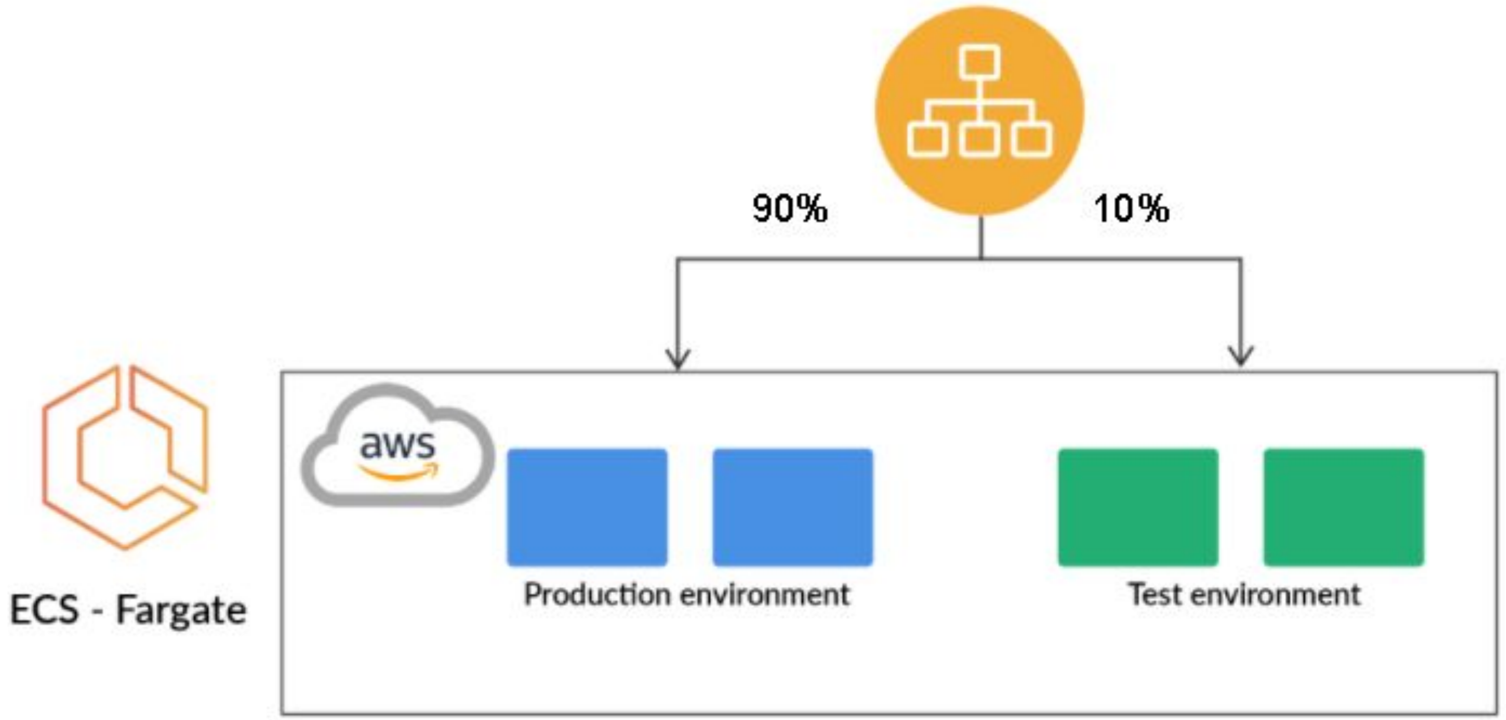
- It is managed by AWS CodeDeploy, a managed deployment service of AWS
- Examples:
    - Required when we want to send 10% of the traffic to the new version and the remaining 90% to the older version.
    - Once verified, you can send more traffic or full traffic to the new version.
    - Not all the users will be impacted if a bug is detected in the new version.

AWS ECS Deployment Type: Blue Green

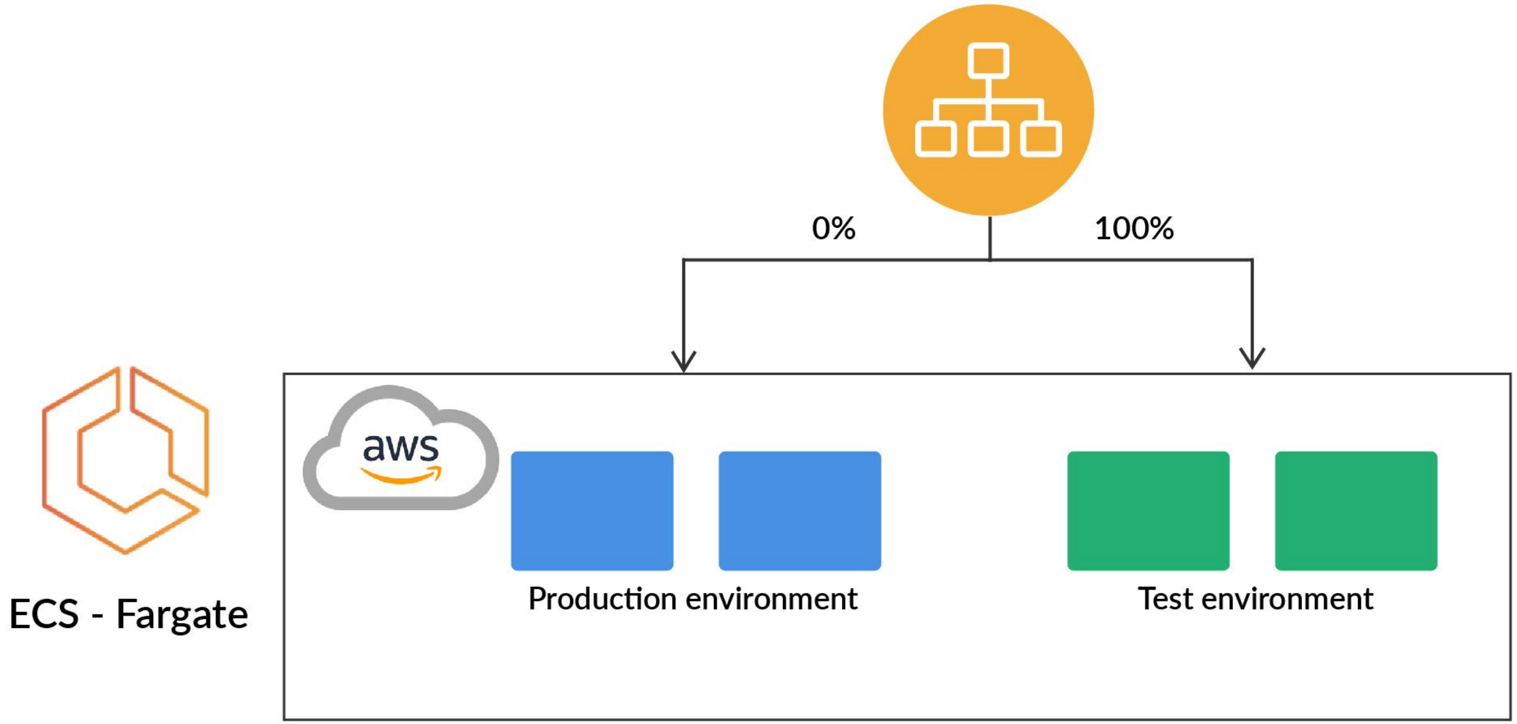


AWS ECS Deployment Type: Blue Green

upGrad



AWS ECS Deployment Type: Blue Green



Demo 2: ECS Service Deployment

- Setting minimumHealthyPercent
- Setting maximumHealthyPercent
- Deploying rolling update in the Voting app

AWS ECS Advanced

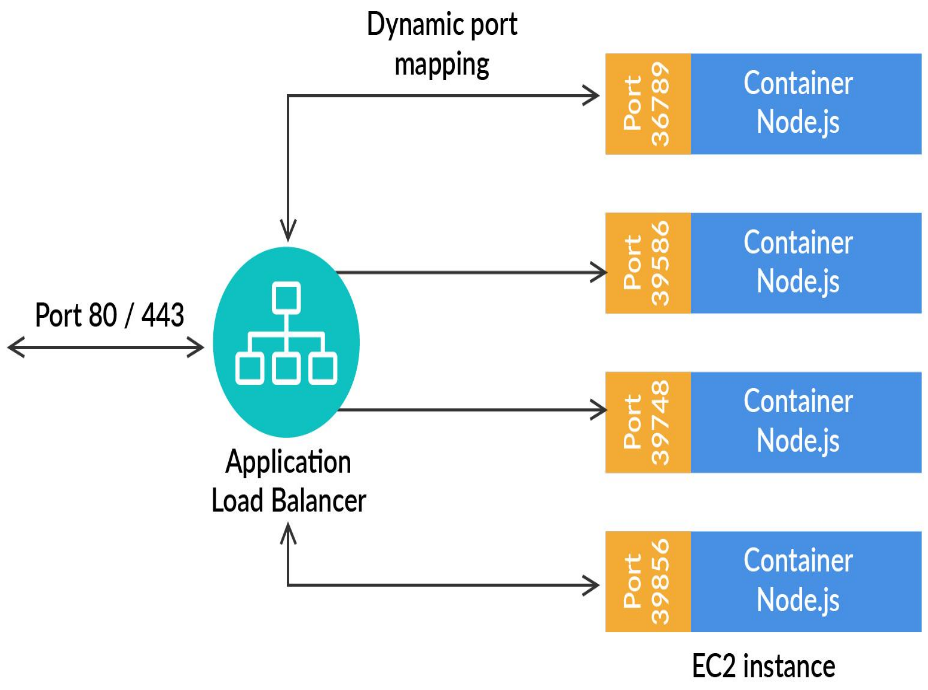
ECS Load Balancing and Autoscaling

AWS ECS Load Balancing

upGrad

In the bridge networking mode:

- You get dynamic port mapping.
- The application load balancer (ALB) finds the right port on the EC2 instances.
- You must allow on the EC2's security group any port from the ALB security group.

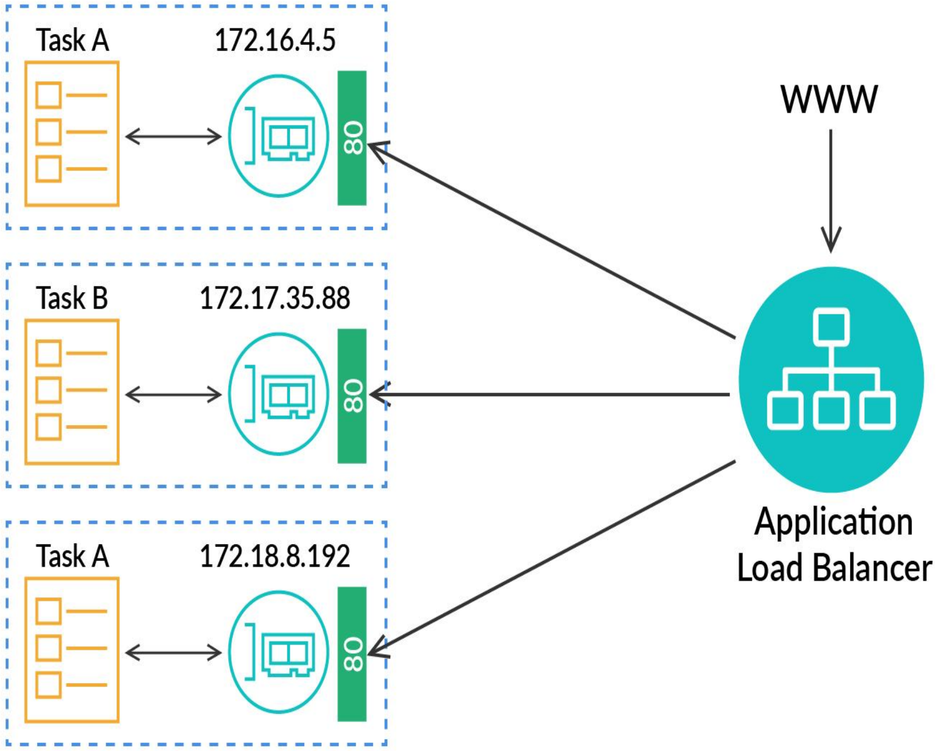


AWS ECS Load Balancing

upGrad

In the ‘awsvpc’ networking mode:

- Each task has its own IP.
- You must allow on the ENI’s security group the task port from the ALB security group.



AWS ECS Autoscaling

- ECS autoscaling includes two types:
  - ECS service autoscaling
    It allows you to increase or decrease the desired count of tasks in your Amazon ECS service automatically as per need.
  - ECS cluster autoscaling
    The number of EC2 instances in an ECS cluster should scale in or out as needed to accommodate new tasks/containers.

AWS ECS Autoscaling: Service

- It gives you the ability to increase or decrease the desired count of tasks in your Amazon ECS service automatically as per need.
- CPU and RAM are tracked in CloudWatch at the ECS service level.
- You can also use other CloudWatch metrics to scale in or out ECS services.
- ECS service autoscaling supports these types of automatic scaling:
  - Target tracking
  - Step scaling
  - Scheduled scaling

AWS ECS Autoscaling: Service

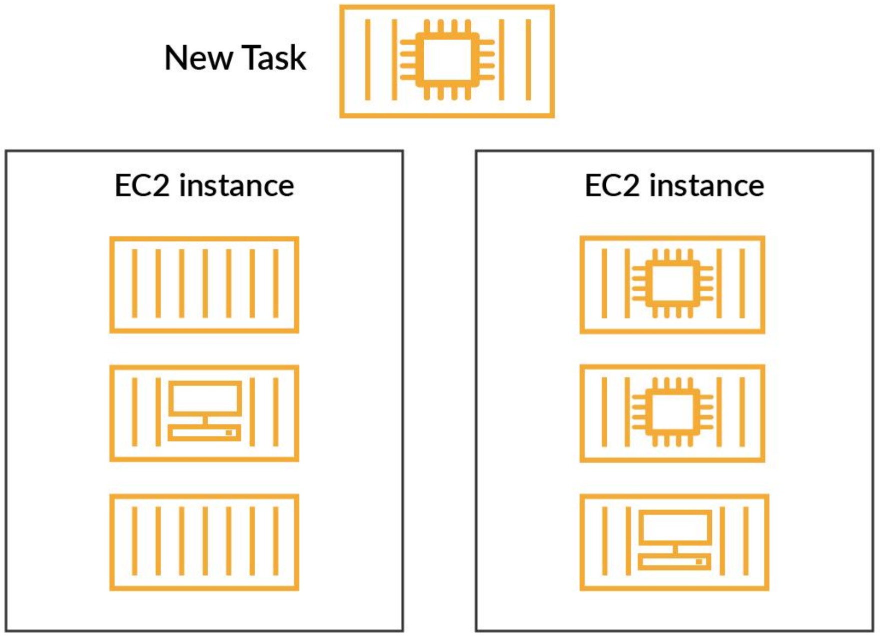
- Target tracking
  - It is based on a target value for a specific metric (CPU/memory).
  - **Example:** Scale in or out to maintain a target CPU threshold of 60%
- Step scaling
  - It varies based on the size of the alarm breach.
  - **Example:** Scale out 1 instance on 50% load. Scale out 1 more on 60%, and scale out 2 instances on reaching 75%
- Scheduled scaling
  - It involves scaling in or out based on a scheduled date and time.
  - **Example:** Scale in 5 instances at 8:00 am IST daily

AWS ECS Autoscaling: Cluster

- The number of EC2 instances in the ECS cluster should scale in or out as needed to accommodate new tasks/containers.
- Why do we need a cluster autoscaler?
    - Many services scale at the same time and there is not enough space to place new tasks.
    - Tasks require a minimum amount of CPU or memory to get placed, but no node is able to fulfil that requirement.
    - You have placement constraints, and there is no node available to fulfil that particular requirement.
    - There is enough resource available on you nodes and you can scale in one or more node by shifting tasks to common nodes.

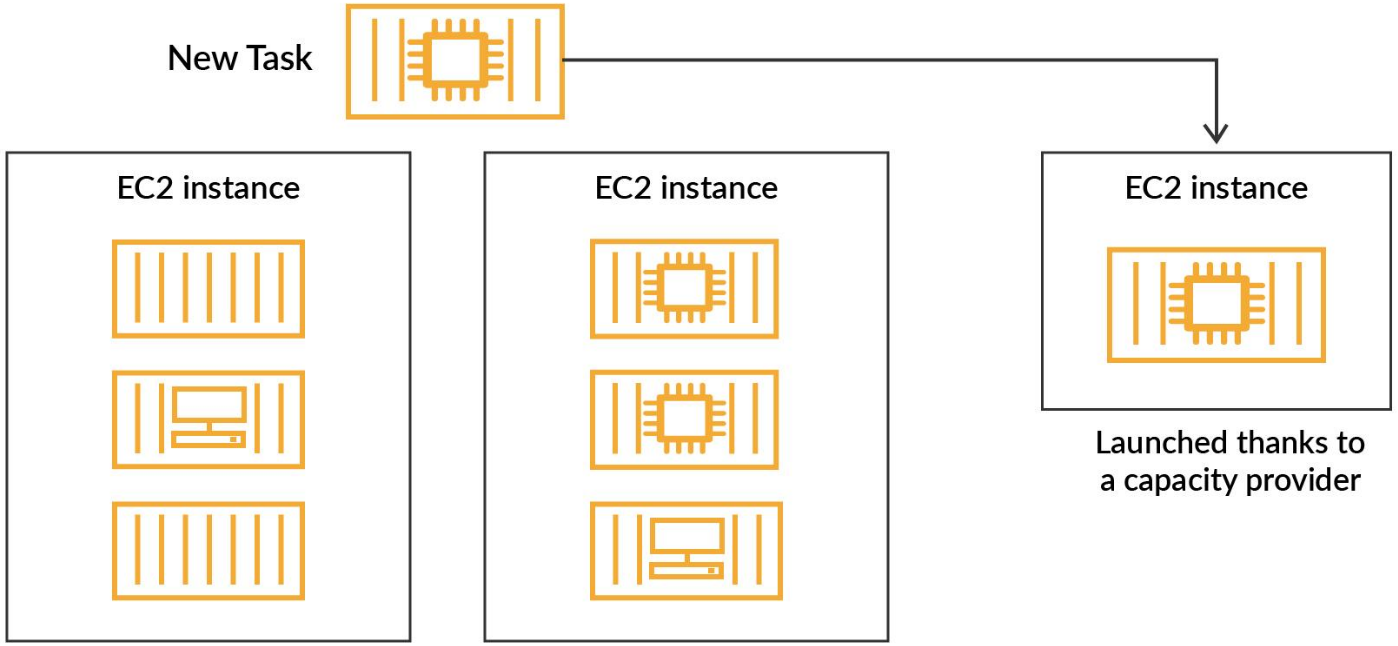
AWS ECS Autoscaling: Cluster

upGrad



Average EC2 CPU < 3%
EC2 instances have no more capacity

AWS ECS Autoscaling: Cluster



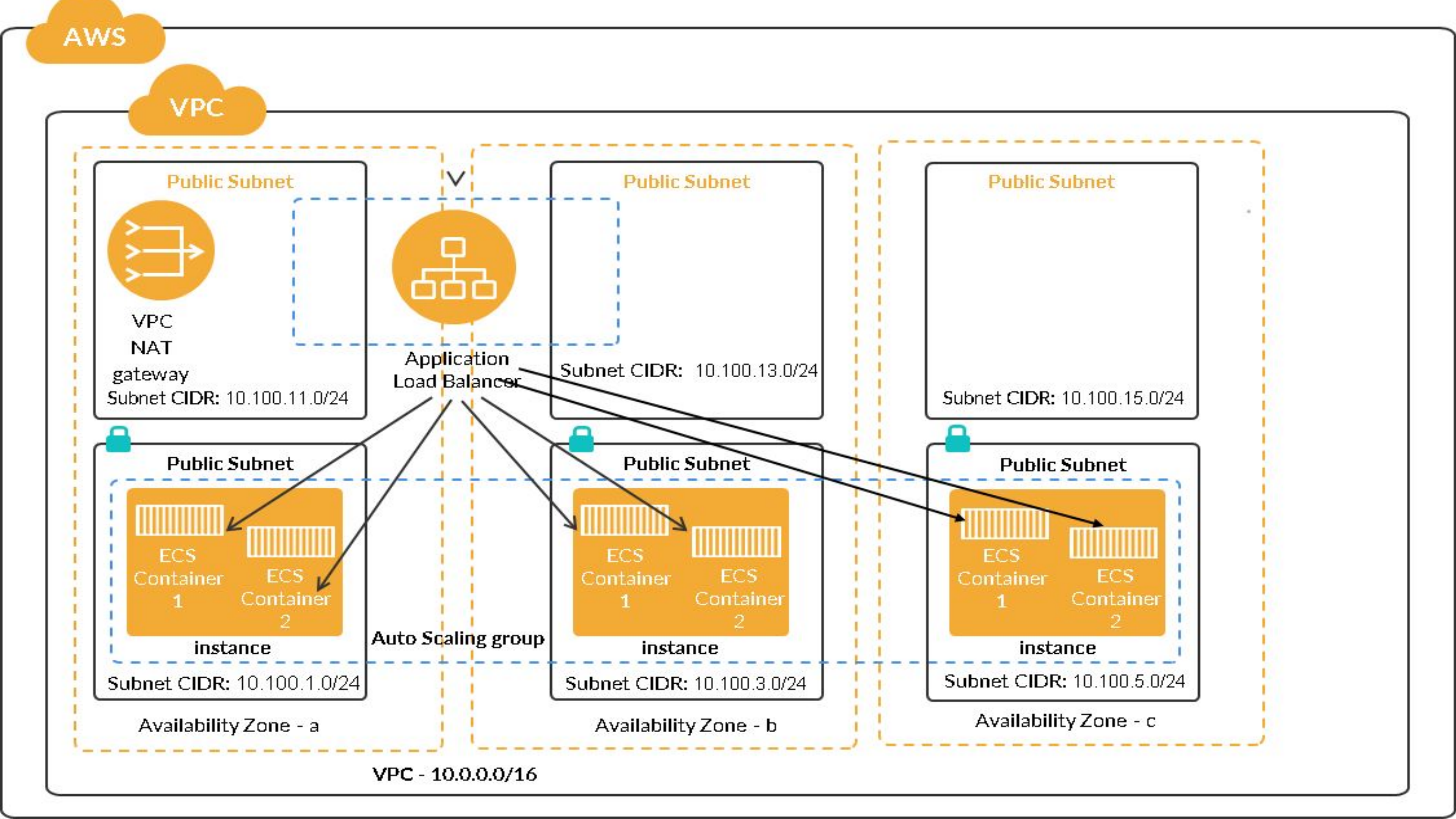
Average EC2 CPU < 3%
EC2 instances have no more capacity

AWS ECS Cluster Autoscaling: Cluster

- A capacity provider is used in association with a cluster to determine the infrastructure that a task runs on:
  - For ECS and Fargate users, the FARGATE and FARGATE_SPOT capacity providers are added automatically.
  - For ECS on EC2, you need to associate the capacity provider with an autoscaling group.

Demo 3: ECS EC2 Mode

- Creating a custom VPC, subnets, IGW, NAT, route tables
- Launch an ECS cluster in EC2 mode
- Launching tasks using the ECS service
- Service discovery using Route53
- Service discovery using ALB for Front-End apps
- Host- and path based routing in ALB
- Cluster autoscaling using Autoscaling Group
- Monitoring and logging
- Verifying the workflow


AWS
VPC
Public Subnet
VPC NAT gateway
Subnet CIDR: 10.100.11.0/24
Application Load Balancer
Public Subnet
Subnet CIDR: 10.100.13.0/24
Public Subnet
Subnet CIDR: 10.100.15.0/24
Public Subnet
ECS Container 1
ECS Container 2
instance
Subnet CIDR: 10.100.1.0/24
Auto Scaling group
Public Subnet
ECS Container 1
ECS Container 2
instance
Subnet CIDR: 10.100.3.0/24
Public Subnet
ECS Container 1
ECS Container 2
instance
Subnet CIDR: 10.100.5.0/24
Availability Zone - a
Availability Zone - b
Availability Zone - c
VPC - 10.0.0.0/16

KPIs for DevOps Engineers

upGrad

Security

- You should avoid using a default VPC and public subnets.
- Only the load balancer should be provisioned in public subnets.
- Strong Network Access Control List (NACL is subnet-level security) and security group rules (instance-level security)
  - No port should be opened to 0.0.0.0/0 (all IPs), except 80 or 443, unless required.
  - Highly restricted or no access to SSH or DB ports like 22, 3306, 5432 and 3389, except the application security group.
- Use of IAM roles over Access keys is highly recommended.
- Use of AWS secrets in Environment values for DB username and passwords.

KPIs for DevOps Engineers

Reliability and Performance

- **Zero-downtime deployment:** Deployment should not have any impact on the end user.
- A proper logging and monitoring setup, along with an alert system, should be in place. (It will be covered in detail in the next module.)
- Autoscaling and load balancing at each component level should be configured:
  - ECS service autoscaling
  - ECS cluster autoscaling
  - **Other components:** RDS, standalone EC2
- The setup should be a multi-AZ setup.

KPIs for DevOps Engineers

Cost

- The ASG should include spot + on-demand EC2 as per requirement.
- Use of reserved instances (available on discount) for predictable workloads.
- There should be no unused EBS volumes/elastic IP in the account.
- You should purge the AMIs and ECR images after the retention period.
- AWS billing alerts should be implemented to receive notifications when cost exceeds the budget (budget described in your policies).
- Tags like Environment, Owner and Team should be placed on every resource so that you can apply filters in the billing console to get more insight on the cause of overbilling.

upGrad

Important Concepts and Questions

Important Questions

1. What is ECS? Explain the different components of the ECS architecture.
2. What are the different pricing models in ECS? Mention some use cases.
3. What are environment variables? How we can you pass an environment variable to the ECS containers?
4. Explain the different types of placement constraints provided by ECS.
5. How can you achieve cluster and service autoscaling in ECS?
6. While building a production application, what are the points that you should consider from a reliability and cost perspective?

upGrad

Doubt Clearance Window

In this class, we covered these topics:

1. Recap of ECS
2. Environment variables
3. ECS IAM roles
4. Task placement
5. Update services and ECS rolling updates
6. ECS load balancing and autoscaling
7. ECS production-ready setup demo:
    a. Custom VPC, public and private subnets, IGW, NAT and route tables
    b. ECS cluster in EC2 mode
    c. Task definition and services
    d. Monitoring, logging, constraints and service discovery
8. Learn the KPIs for DevOps engineers

Homework

1. Deploy a web application of your choice on the ECS Optimised image t2.small instance and configure blue green deployment on it.
2. Set up upGrad's application on AWS ECS EC2 mode:
    a. Use a custom VPC.
    b. Create two public and two private subnets, along with IGW, NAT and route tables.
    c. Create an ECS cluster and set up an application on the EC2 mode in private subnets.
    d. Launch ALB in the public subnet and access the application on port 80 of ALB.
    e. Set up proper monitoring and logging.
    f. Use route 53 for service discovery and try accessing the application using the DNS from one application to other.

upGrad

Tasks to Complete After Today's Session

MCQs
Conceptual Questions
Project Checkpoint

upGrad

#RahoAmbitious

Thank You!