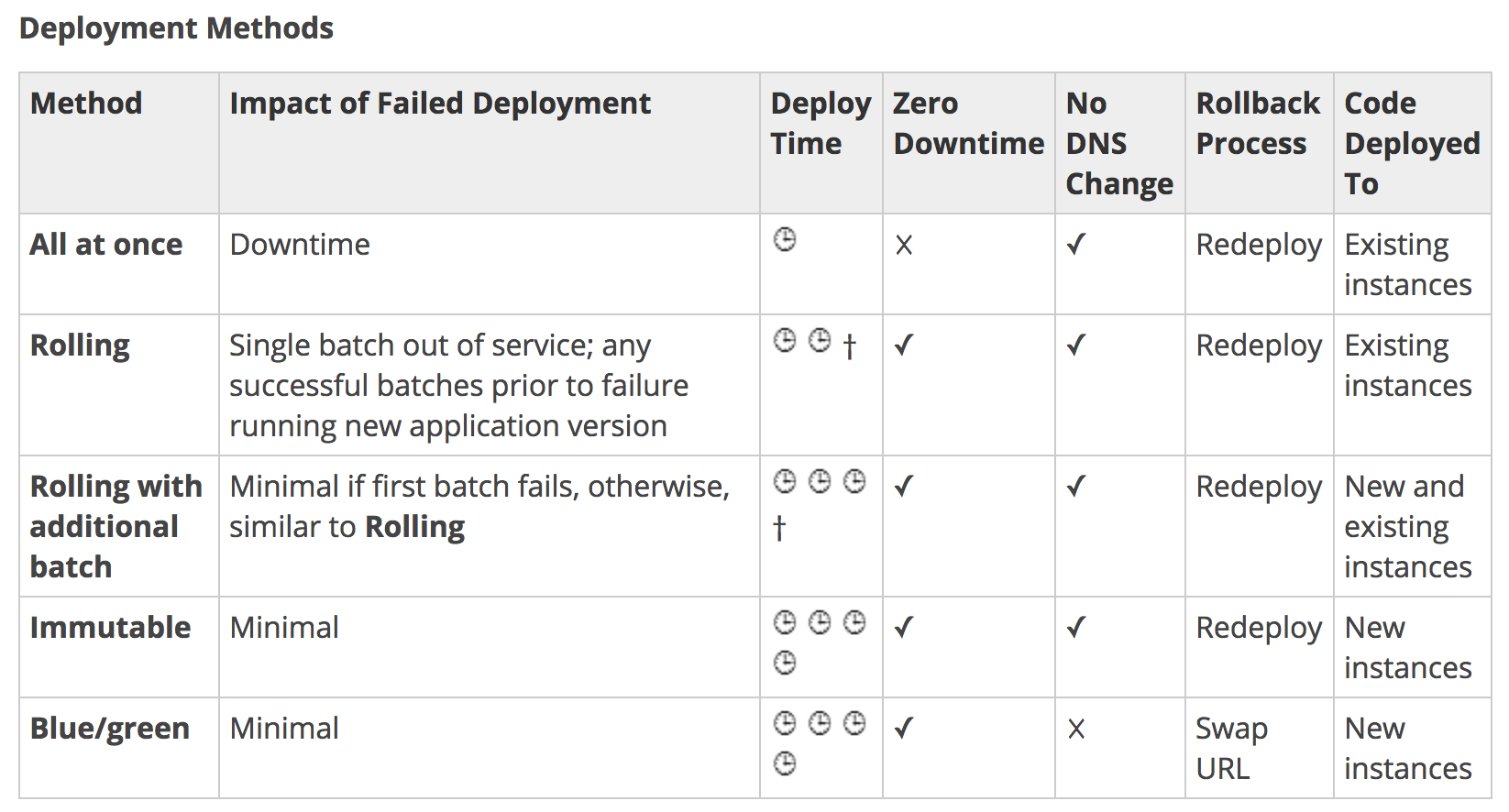
AWS Elastic Beanstalk Deployment Strategies

# AWS Elastic Beanstalk Deployment Strategies

* Elastic Beanstalk supports environments as
  + Single Instance environments, with a single instance and auto scaling to maintain the minimum/maximum 1 instance
  + Load Balanced environments, with load balancing and auto scaling
* Elastic Beanstalk allows multiple deployment options or strategies that can be selected depending upon the requirements for deployment time, downtime, DNS change and rollback process

#### **NOTE: Advanced Topic required for DevOps Professional Exam Only**

## Elastic Beanstalk Deployment Methods



### All at Once Deployments

* Elastic Beanstalk environment uses all-at-once deployments if it is created with a different client (API, SDK, or AWS CLI)
* All at Once deployments performs an in place deployment on all instances at the same time
* All at Once deployments are simple and fast, however rollback would take time in case of any issues

### Rolling Deployments

* Elastic Beanstalk environment uses rolling deployments if it is created with console or EB CLI
* Elastic Beanstalk splits the environment’s EC2 instances into batches and deploys the new version of the application to one batch at a time, leaving the rest of the instances in the environment running the old version
* During a rolling deployment, part of the instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.
* Elastic Beanstalk performs the rolling deployments as
  + When processing a batch, detaches all instances in the batch from the load balancer, deploys the new application version, and then reattaches the instances.
  + To avoid any connection issues when the instances are detached, connection draining can be enabled on the load balancer
  + After reattaching the instances in a batch to the load balancer, ELB waits until they pass a minimum number of health checks (the Healthy check count threshold value), and then starts routing traffic to them.
  + Elastic Beanstalk waits until all instances in a batch are healthy before moving on to the next batch.
  + When all instances in the batch pass enough health checks to be considered healthy by ELB, the batch is complete.
  + If a batch of instances does not become healthy within the command timeout, the deployment fails.
  + If a deployment fails after one or more batches completed successfully, the completed batches run the new version of the application while any pending batches continue to run the old version.
  + If the instances are terminated from the failed deployment, Elastic Beanstalk replaces them with instances running the application version from the most recent successful deployment.

### Rolling with Additional Batch Deployments

* Rolling with Additional Batch deployments are helpful when you need to maintain full capacity during deployments
* This deployment is similar to Rolling deployments, except they do not do an in place deployment but a disposable one, launching a new batch of instances prior to taking any instances out of service
* When the deployment completes, Elastic Beanstalk terminates the additional batch of instances.
* Rolling with additional batch deployment does not impact the capacity and ensures full capacity during the deployment process

### Immutable Deployments

* All at Once and Rolling deployment method updates existing instances.
* If you need to ensure the application source is always deployed to new instances, instead of updating existing instances, environment can be configured to use [immutable updates](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environmentmgmt-updates-immutable.html) for deployments.
* Immutable updates are performed by launching a second Auto Scaling group is launched in the environment and the new version serves traffic alongside the old version until the new instances pass health checks.
* Immutable deployments can prevent issues caused by partially completed rolling deployments. If the new instances don’t pass health checks, Elastic Beanstalk terminates them, leaving the original instances untouched.

### Blue Green Deployments

* Elastic Beanstalk performs an in-place update when application versions is updated, that may result in application becoming unavailable to users for a short period of time
* Blue Green approach is suitable for deployments that depend on resource configuration changes or a new version that can’t run alongside the old version
* Elastic Beanstalk enables the Blue Green deployment through **Swap Environment URLs**feature
* **Blue Green deployment provides almost zero downtime solution, where a new version is deployed to a separate environment, and then CNAMEs of the two environments are swapped to redirect traffic to the new version**
* **Blue/green deployments require that the environment runs independently of the production database i.e. not maintained by Elastic Beanstalk, if your application uses one. Because if the environment has an RDS DB instance attached to it, the data will not transfer over to the second environment, and will be lost if the original environment is terminated**
* Blue Green deployment entails a DNS change and hence do not terminate the old environment until the DNS changes have been propagated and the old DNS records expire.
* DNS servers do not necessarily clear old records from their cache based on the time to live (TTL) you set on the DNS records.

## AWS Certification Exam Practice Questions

1. When thinking of AWS Elastic Beanstalk, the ‘Swap Environment URLs’ feature most directly aids in what? [CDOP]
   1. Immutable Rolling Deployments
   2. Mutable Rolling Deployments
   3. Canary Deployments
   4. **Blue-Green Deployments**(Simply upload the new version of your application and let your deployment service (AWS Elastic Beanstalk, AWS CloudFormation, or AWS OpsWorks) deploy a new version (green). To cut over to the new version, you simply replace the ELB URLs in your DNS records. Elastic Beanstalk has a Swap Environment URLs feature to facilitate a simpler cutover process.)
2. You need to deploy a new version of your application. You’d prefer to use all new instances if possible, but you cannot have any downtime. You also don’t want to swap any environment URLs. You’re running t2.large instances and you normally need 15 instances to meet capacity. Which deployment method should you use? Choose the correct answer:
   1. Rolling Updates
   2. Blue/Green
   3. **Immutable**
   4. All at Once
3. Your team is responsible for an AWS Elastic Beanstalk application. The business requires that you move to a continuous deployment model, releasing updates to the application multiple times per day with zero downtime. What should you do to enable this and still be able to roll back almost immediately in an emergency to the previous version? [CDOP]
   1. Enable rolling updates in the Elastic Beanstalk environment, setting an appropriate pause time for application startup.
   2. **Create a second Elastic Beanstalk environment running the new application version, and swap the environment CNAMEs.**
   3. Develop the application to poll for a new application version in your code repository; download and install to each running Elastic Beanstalk instance.
   4. Create a second Elastic Beanstalk environment with the new application version, and configure the old environment to redirect clients, using the HTTP 301 response code, to the new environment.