Question 1: Explain the differences between development, integration, staging, and production environments.

Answer:

Strategy	How it works	Advantages	Disadvantages	When to use	Links
Recreate deployment	Basically, before creating new instances with updated configurations, it shut downs all existing instances provoking a temporary runtime.	It ensures a new and clean environment, which will be exactly what was in the new pods.	As mentioned before, it's impossible to do a transition without a downtime, meaning the instances/features can not be updated without shutting down them.	Best use case is when the nature of the instance allows to work in batchs and doesn't need a totally continuous service. Then, it allows to shutdown the system to create the new environment.	Link
Rolling Update	Works by updating each post such that the system never shuts down. Basically, it redirects traffic out of the pod that needs to be updated making the other pods to handle more requests. When the pod is done updating, it will start handling requests while another pod takes the turn to update.	Keeps availability of the system, it allows a gradual introduction of a new version making easier to keep track of any issues that may come.	Rolling back might be challenging because you'll need to rollback certain instances, not the whole environment. Also, the service operates in inconsistent states while updating all pods.	When having a service that needs continuous availability. This strategy allows to keep the service running while in the back, updates are happening.	Link Link Link
Blue-Green deployment	You have two environments: Blue (old one) and Green (new one with the new version). After setting up the green environment, need to redirect all traffic to the new environment and keep track of it's performance. If something wrong happens and need to rollback, you can switch back the traffic to the blue environment.	Since it relies only in traffic redirection, it allows a very high availability. It also allows a very fast rollback process (also allows to deploy with more confidence).	Two environments running means almost double cost of direct operating costs. Also, needs more overhead to keep track of performance of new environment. The load balancer is also a thing, to know when and how to switch.	For organizations or services with a very high demand for availability. Also, it's good for regulated companies that need to be compliant and need transparency when doing changes to the platform.	Link
Canary deployment	Works by releasing a new version only to a subset of all users. It's not a all-in upgrade, but a very partial one.	It mitigates risk by rolling out the updates to a subset of users. It keep the update controlled, and if it fails, it will only do so for a small subset of all users. It's cost effective since it doesnt need two environments to run	It doesn't catch all problems since its only targeted to a subset of users, which may not be handling the service as the total user population. It need a strategy for which users the new updates will be rolled out.	Very useful for testing technical stability of new features, benchmark performance, and others.	Link

Strategy	How it works	Advantages	Disadvantages	When to use	Links
A/B Testing	Like Canary Deployment, it releases new features to a subset of all users. It works by releasing and then record performance of the old v/s the new performance. After it, management can decide with data in hand if developing completely or deploying completely the new feature. It's mostly related with UI of the service or the app.	Cost effective to check value of new features. Provides real data on what happened with the new version vs the old version.	Need strategy of redirection. It doesn't provide any explanation on why things happened with the new version, but only *what happened*.	Most of the times use it to try UI things. At my old job we used it a lot to try new wording for a message we wanted to communicate, pictures, colors, etc.	Link

For the next questions:

- **Refer to Github Repo to see all files.**

Question 2: Rolling Update Implementation

- Get deployments/pods showing initial state

```
get deployments/pods showing initial state
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
roll-update-app   3/3     3            3           9m39s
NAME                                 READY   STATUS    RESTARTS   AGE
roll-update-app-7b7546747c-24z9l   1/1     Running   0          89s
roll-update-app-7b7546747c-4h6f8   1/1     Running   0          101s
roll-update-app-7b7546747c-d4sdv   1/1     Running   0          101s
```

Rollout status during update

```
rollout status during update...
deployment.apps/roll-update-app image updated
```

Get pods showing successful update

```
get pods showing successful update...
Waiting for deployment "roll-update-app" rollout to finish: 0 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "roll-update-app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "roll-update-app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "roll-update-app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "roll-update-app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "roll-update-app" rollout to finish: 2 of 3 updated replicas are available...
Waiting for deployment "roll-update-app" rollout to finish: 2 of 3 updated replicas are available...
deployment "roll-update-app" successfully rolled out
```

Rollout undo command and result

```
rollout undo command and result
deployment.apps/roll-update-app rolled back
```

Curl or browser showing the service is accessible

```
Listing updated Pods...
NAME                              READY   STATUS              RESTARTS   AGE
roll-update-app-679f4fb495-6tc9k   1/1     Running             0          25s
roll-update-app-679f4fb495-7q24m   1/1     Running             0          25s
roll-update-app-679f4fb495-wjr9r   1/1     Terminating         0          12s
roll-update-app-7b7546747c-lqbjn   0/1     ContainerCreating   0          0s
roll-update-app-7b7546747c-r4gkh   0/1     Pending             0          0s
Checking Service details...
NAME                 TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
roll-update-service   NodePort   10.105.143.223   <none>        8080:32297/TCP   9m14s
```

NAMESPACE	NAME	TARGET PORT	URL
default	roll-update-service	8080	http://192.168.49.2:32297

```
🔗  Starting tunnel for service roll-update-service.
```

NAMESPACE	NAME	TARGET PORT	URL
default	roll-update-service		http://127.0.0.1:57651

```
🎉  Opening service default/roll-update-service in default browser...
```

Question 3: Blue-Green Deployment

- Showing blue deployment running

```
------------------------------
Showing blue deployment running
deployment.apps/blue unchanged
deployment "blue" successfully rolled out
------------------------------
```

Showing both blue and green running

```
---------------------------------
Showing green deployment running
deployment.apps/green unchanged
deployment "green" successfully rolled out
---------------------------------
```

Describe service showing blue selector

```
--------------------------------
Describe service showing blue selector
Name:                     blue-green-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=app,version=blue
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.52.22
IPs:                      10.97.52.22
Port:                     <unset>  8080/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30246/TCP
Endpoints:                10.244.0.26:80,10.244.0.27:80
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
--------------------------------
```

Commands switching service to green

```
------------------------------
Commands switching service to green
Commands are: kubectl patch svc blue-green-service -p '{"spec":{"selector":{"app":"app","version":"green"}}}'
service/blue-green-service patched
------------------------------
```

Describe service showing green selector

```
--------------------------------
Describe service showing green selector
Name:                     blue-green-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=app,version=green
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.52.22
IPs:                      10.97.52.22
Port:                     <unset>  8080/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30246/TCP
Endpoints:                10.244.0.29:80,10.244.0.30:80
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
--------------------------------
```

Execution of rollback script and verification

```
--------------------------------
Execution of rollback script and verification
service/blue-green-service patched
Describe service to check it went back to blue
Name:                     blue-green-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=app,version=blue
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.97.52.22
IPs:                      10.97.52.22
Port:                     <unset>  8080/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30246/TCP
Endpoints:                10.244.0.26:80,10.244.0.27:80
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
--------------------------------
```

Question 4: Canary Deployment

- **For the last part of rebalancing load, I used Copilot to try to debug it, but it always showed 80/20 relation, even when I patched it.**

- Showing stable deployment running

```
------------------------------
Showing stable deployment running
deployment.apps/stable-deployment unchanged
deployment "stable-deployment" successfully rolled out
------------------------------
```

Showing both stable and canary running

```
--------------------------------
Showing canary deployment running
deployment.apps/canary-deployment unchanged
deployment "canary-deployment" successfully rolled out
--------------------------------
```

Describe showing initial traffic split

```
--------------------------------
Describe showing initial traffic split
Name:             canary-ingress
Labels:           <none>
Namespace:        default
Address:          192.168.49.2
Ingress Class:    nginx
Default backend:  <default>
Rules:
  Host         Path  Backends
  ----         ----  --------
  myapp.local
               /   canary-service:80 (10.244.0.44:80,10.244.0.45:80)
Annotations:   nginx.ingress.kubernetes.io/canary: true
               nginx.ingress.kubernetes.io/canary-weight: 20
Events:
  Type    Reason  Age                  From                      Message
  ----    ------  ----                 ----                      -------
  Normal  Sync    2m5s (x2 over 3m3s)  nginx-ingress-controller  Scheduled for sync
--------------------------------
```

Multiple curl requests showing ~20% going to canary

```
----------------------------------
Multiple curl requests showing ~20% going to canary
Traffic distribution:
Stable: 796
Canary: 204
----------------------------------
```

After promotion showing 50/50 traffic split

```
  ----    -------  ----                 ----                      -------
  Normal  Sync    11s (x54 over 35m)  nginx-ingress-controller  Scheduled for sync
Testing traffic after 50/50 promotion...
Traffic distribution:
Stable: 800
Canary: 200
----------------------------------
```

Final state with 100% traffic to canary

```
--------------------------------
Promoting canary to 100% traffic...
ingress.networking.k8s.io/canary-ingress patched
Testing traffic after 100% promotion...
Traffic distribution:
Stable: 781
Canary: 219
--------------------------------
```