

Explain Blue-Green Deployment Technique

Answer

**Blue-green deployment** is a technique that reduces downtime and risk by running two identical production environments called Blue and Green. At any time, only one of the environments is live, with the live environment serving all production traffic. For this example, Blue is currently live and Green is idle.

As you prepare a new version of your software, deployment and the final stage of testing takes place in the environment that is not live: in this example, Green. Once you have deployed and fully tested the software in Green, you switch the router so all incoming requests now go to Green instead of Blue. Green is now live, and Blue is idle.

This technique can eliminate downtime due to application deployment. In addition, blue-green deployment reduces risk: if something unexpected happens with your new version on Green, you can immediately roll back to the last version by switching back to Blue.

## What's the difference between a blue/green deployment and a rolling deployment?

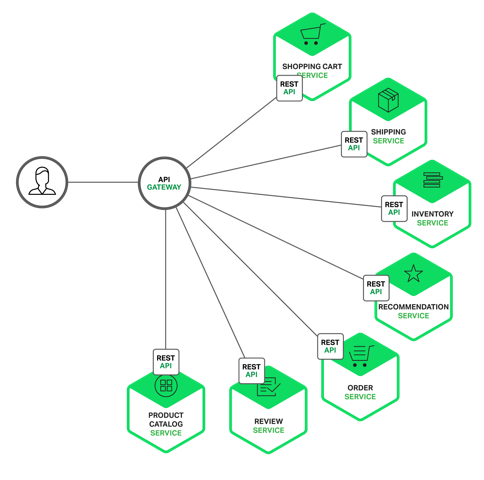
Answer

* In **Blue Green Deployment**, you have TWO complete environments. One is Blue environment which is running and the Green environment to which you want to upgrade. Once you swap the environment from blue to green, the traffic is directed to your new green environment. You can delete or save your old blue environment for backup until the green environment is stable.
* In **Rolling Deployment**, you have only ONE complete environment. The code is deployed in the subset of instances of the same environment and moves to another subset after completion.

## Explain what is the API Gateway pattern

Answer

An **API Gateway** is a server that is the single entry point into the system. It is similar to the Facade pattern from object‑oriented design. The API Gateway encapsulates the internal system architecture and provides an API that is tailored to each client. It might have other responsibilities such as authentication, monitoring, load balancing, caching, request shaping and management, and static response handling.



A major benefit of using an API Gateway is that it encapsulates the internal structure of the application. Rather than having to invoke specific services, clients simply talk to the gateway.

**Pros**:

* Freedom to use different technologies
* Each microservices focuses on single capability
* Supports individual deployable units
* Allow frequent software releases
* Ensures security of each service
* Mulitple services are parallelly developed and deployed

**Cons**:

* Increases troubleshooting challenges
* Increases delay due to remote calls
* Increased efforts for configuration and other operations
* Difficult to maintain transaction safety
* Tough to track data across various boundaries
* Difficult to code between services