Deployment Strategies

CSSE6400

Richard Thomas

May 23, 2022

Definition 1. Deployment Strategy

How a software system is made available to clients.

Deployment Strategies

- Branching Strategies
- Recreate Deployment
- Rolling Deployment
- Blue/Green Deployment
- Canary Deployment
- A/B Deployment
- Shadow Deployment

There isn't any one perfect deployment strategy.

Definition 2. Branching

Copying the trunk to allow separate and parallel development.

- Branches deviate from the trunk.
- A few different branching strategies.

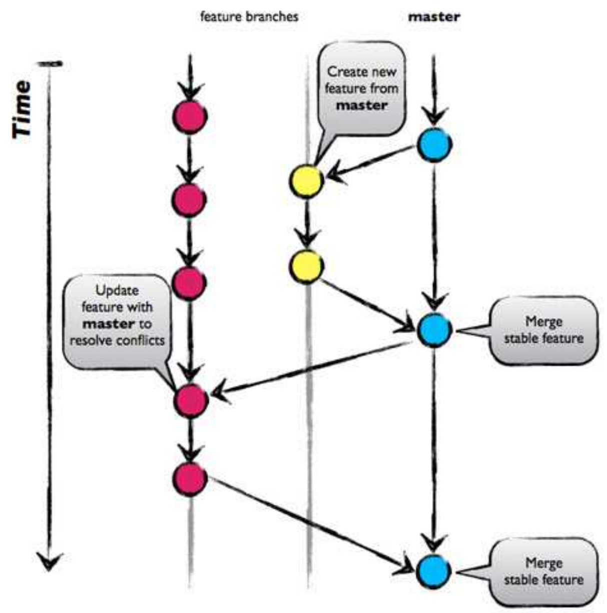
Branching Strategies

- GitHub Flow
- GitLab Flow
- Release Branches

Branching strategies supporting deployment strategies.

GitHub Flow [1]

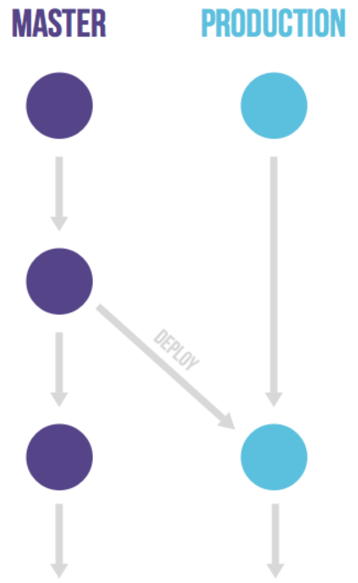
- Main is always deployable
- Create branch
- Make changes
- Create pull request
- Resolve issues
- Merge pull request
- Delete branch


feature branches
master
Time
Create new feature from **master**
Update feature with **master** to resolve conflicts
Merge stable feature
Merge stable feature

- Supports CI & CD.
- Expects there is a single deployable version (e.g. cloud / web systems).

GitLab Flow [2]

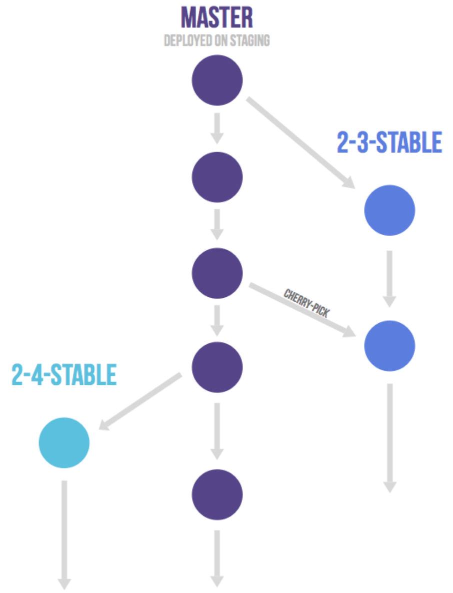
- Supports deployment windows
  - Merge to production
  - Deploy when allowed
- Production branch
  - Plus alpha, beta, …
- Still have
  - Feature branches
  - Pull requests



- Deployment windows examples:
  - App store approval
  - Server availability
  - Support availability

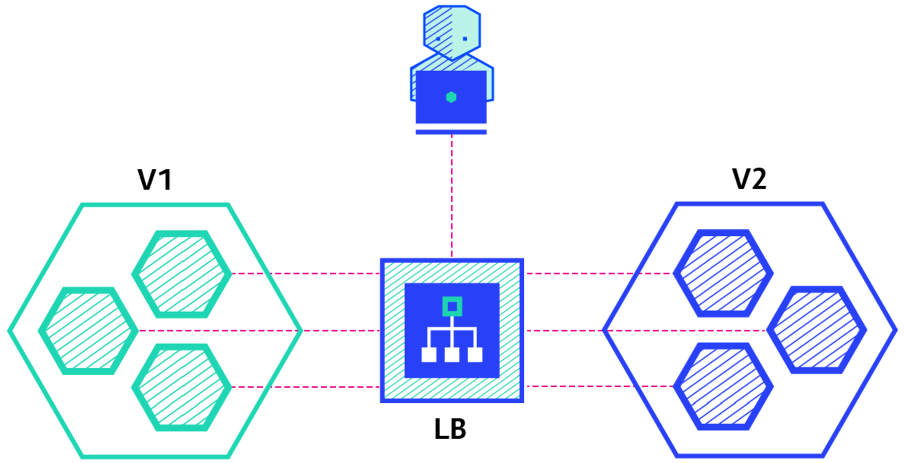
Release Branches [2]

- Supports multiple versions of system
- Feature development in main
- Released versions are branches
- Bug fixes in main
  - Cherry-pick into branches



- Cherry-pick: commit is copied from one branch to another, but the branches aren't merged.

Recreate Deployment [3]



- Shutdown version 1.
- Deploy version 2.
- Requires downtime.

Recreate Deployment

Pros

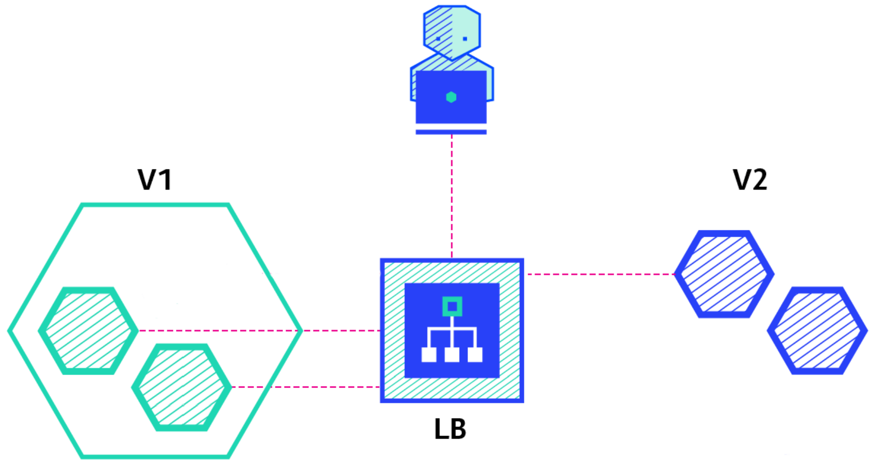
- Easy
- Renewed state
    - App reinitialised
    - Persistent storage consistent with system version

Cons

- Downtime

Renewed state means app is reinitialised and db is consistent with system version.

Rolling Deployment [3]



- Slowly roll out new version.
- Pool of instances of v1 behind load balancer.
- Deploy an instance of v2.
- Add v2 instance to pool.
- Remove one v1 instance from pool.
- Continue until v2 is fully deployed, replacing v1.

Rolling Deployment

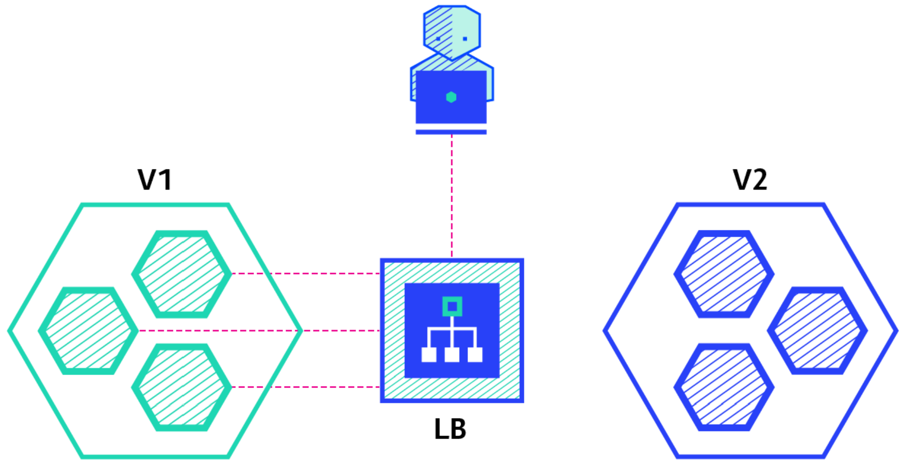
Pros

- Fairly easy
- Slow release of new version
  - Observe issues
  - Rollback
- Stateful instances can finish gracefully
  - Instance is killed when inactive

Cons

- Time
- Need to support multiple APIs
- No control over traffic to different versions

Blue-Green Deployment [3]



- V2 deployed alongside v1, including same number of instances.
- V2 tested in production environment.
- Load balancer switched to use v2 instances
- Shutdown v1 instances.

Blue-Green Deployment

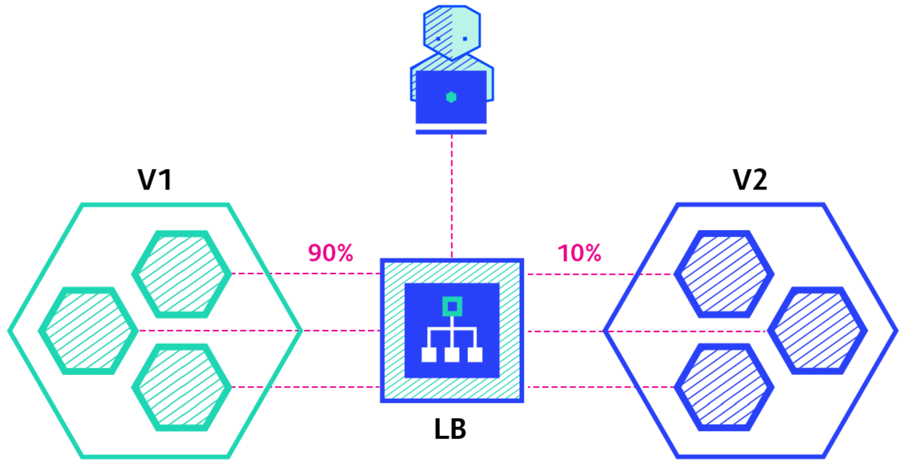
Pros

- Instant release of new version
- Fast rollback if necessary
- Only one version ‘live’ at any time
  - No versioning conflicts

Cons

- Expensive
  - Double the infrastructure
- Stateful instance version switch difficult
  - Can’t kill instance in middle of a transaction

Canary Deployment [3]



- Gradually shift traffic from v1 to v2.
- Traffic usually split by percent (e.g. 90/10, 80/20, ...).
- Allows a trial deployment to see what happens.

Pros

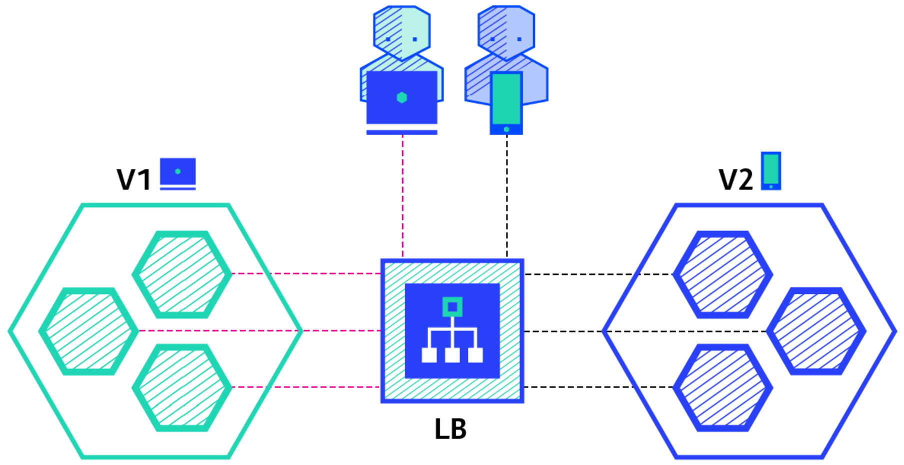
- New version released to subset of users
- Can monitor perform-ance and error rates
- Easy and fast rollback

Cons

- Slow
- Often implies poor testing

Canary is commonly used to see if something works or will fail in production.

A/B Deployment [3]



- Actually it's A/B Testing.
- Both versions are deployed and usage evaluated, usually via analytics.
- Deploy the version that has best usage result.

A/B Deployment

Pros

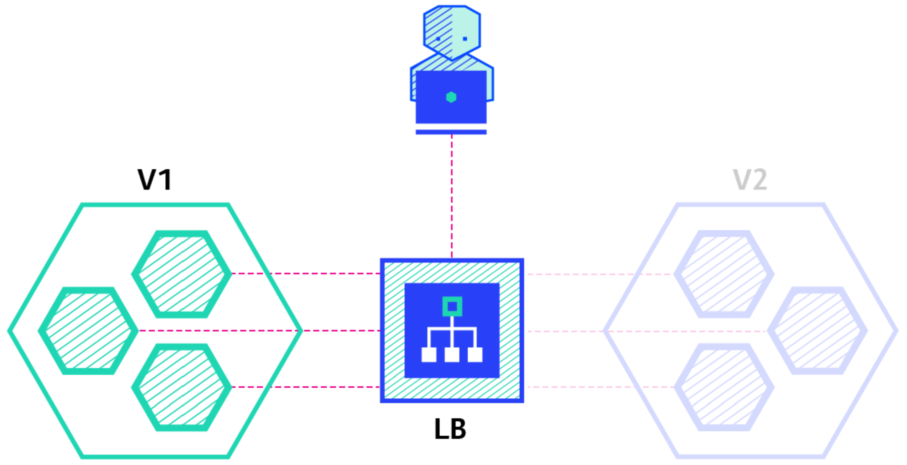
- Multiple versions run in parallel
- Full control over traffic distribution

Cons

- Needs intelligent load balancer
- Debugging a version is difficult
  - Need good logs & tools

A/B testing & deployment requires sophisticated infrastructure and analytics to do well.

Shadow Deployment [3]



- Complex to setup.
- V2 deployed alongside v1.
- All traffic is sent to v1 & v2.
- Tests v2 ability to handle production load.
- Doesn’t impact on production traffic or user experience.
- V2 rolled out when it demonstrates it is stable.
- Need to manage interactions with external services (e.g. payment gateway).
- When customer checks out their shopping cart, you don’t want to send two payment requests from v1 & v2.
- Mock external services.
- Persistent data from v1 (production data) needs to be copied to v2 when it’s deployed as production, with any data migration.

Shadow Deployment

Pros

- Performance testing with production traffic
- No impact on users

Cons

- Expensive
  - Double the infrastructure
- Complex to setup
  - Need mocks for external services

Performance testing may give false confidence – it’s not user testing.

Deployment Strategy Options

- Staging or beta testing
  - Recreate or Rolling
- Production (Live)
  - Rolling or Blue/Green
- Uncertain of system stability
  - Canary
- Evaluation
  - A/B or Shadow

There isn’t any one perfect deployment strategy.

Deployment Considerations [3]

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't	✓	✓	✗	■■■	□□□	□□□	■■■

References

[1] Rowan Haddad.
What are the best git branching strategies.
https://www.flagship.io/git-branching-strategies/, March 2022.

[2] Introduction to gitlab flow.
https://repository.prace-ri.eu/git/help/topics/gitlab_flow.md.

[3] Etienne Tremel.
Six strategies for application deployment.
https://thenewstack.io/deployment-strategies/, November 2017.