Deploy as you need

Karen Hernández González

Senior IT Infrastructure Engineer at Nu Mx

- InfraOps, DevOps, SRE roles
- Photography, wine, learn new things and SW
- Automation, CI/CD Topics, IaC

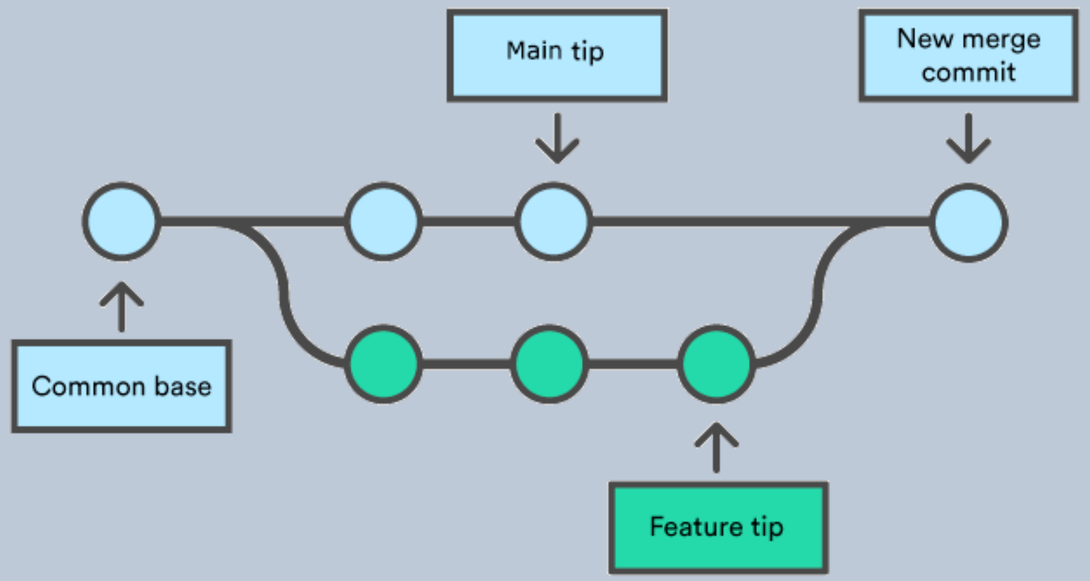
in: @akarenhdez
Tw: @Karen_Heez

Basic Concepts

- **Pipeline:** set of automated processes and tools that allows developers and operations professionals to collaborate on building and deploying code to a production environment.

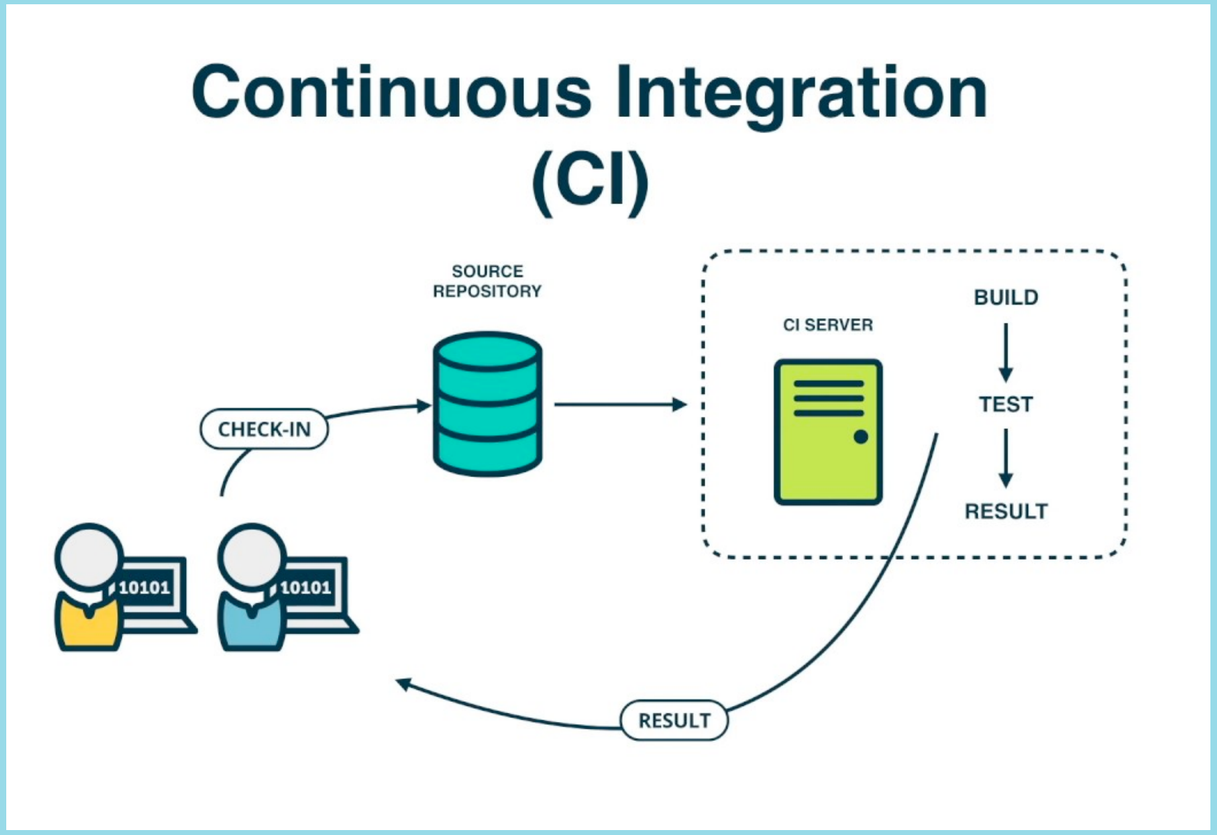
- **Build:** The build stage is the first stretch of a CI/CD pipeline, and it automates steps like downloading dependencies, installing tools, and compiling. Besides building code, build automation includes using tools to check that the code is safe and follows best practices.

- **Merge:** Merging is Git's way of putting a forked history back together again.



Continuous integration vs. delivery vs. deployment

Continuous Integration



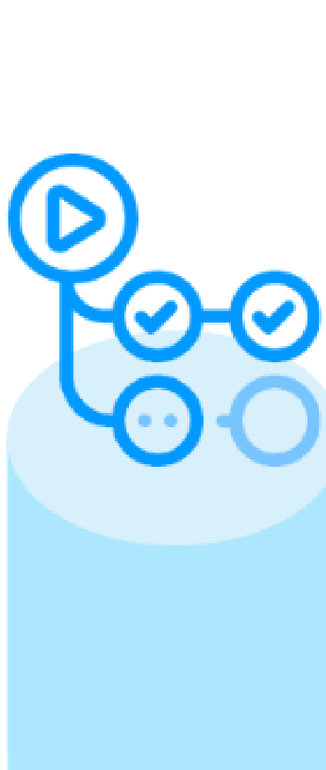
It is the practice of integrating changes from different developers in the team into a mainline as early as possible, in best cases several times a day. This makes sure the code individual developers work on doesn't divert too much. When you combine the process with automated testing, continuous integration can enable your code to be dependable.

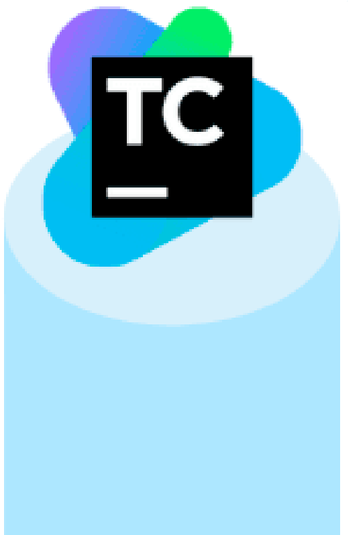
What you need:

- Automated tests (more time to write them)
- Build server (Gitlab CI, Jenkins, Circle CI)
- Merge changes as often as possible

What you gain

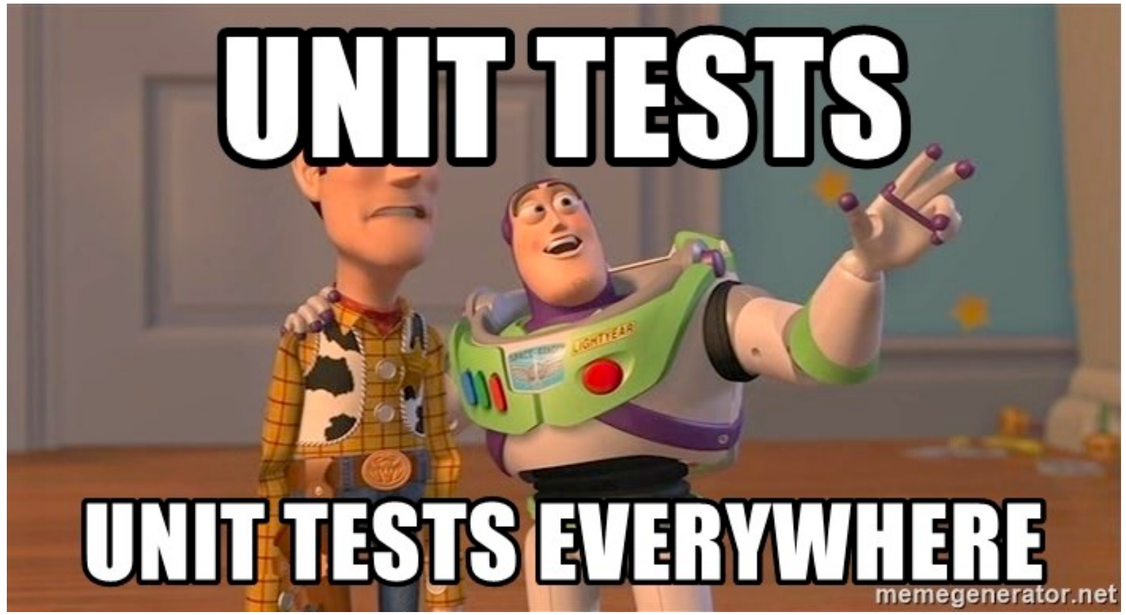
- Less bugs
- Easy building and fast feedback
- Many tests at the same time



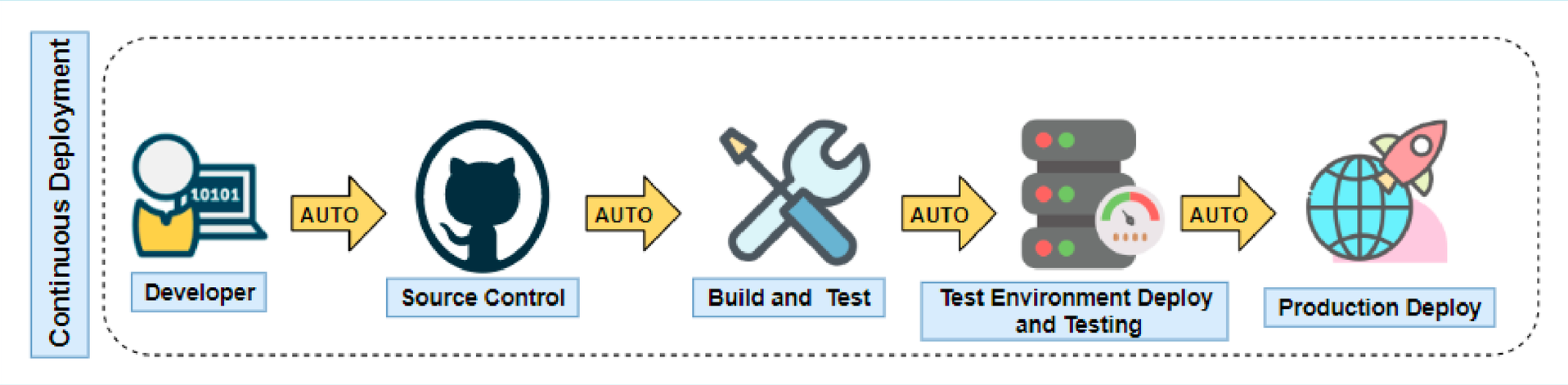


Types of tests

- Unit tests.
- Integration tests
- Functional tests
- End-to-end tests
- Acceptance testing
- Performance testing
- Smoke testing



Continuous Delivery



It is closely related to continuous integration and refers to keeping your application deployable at any point. It involves frequent, automated deployment of the main branch to a production environment following automated testing.

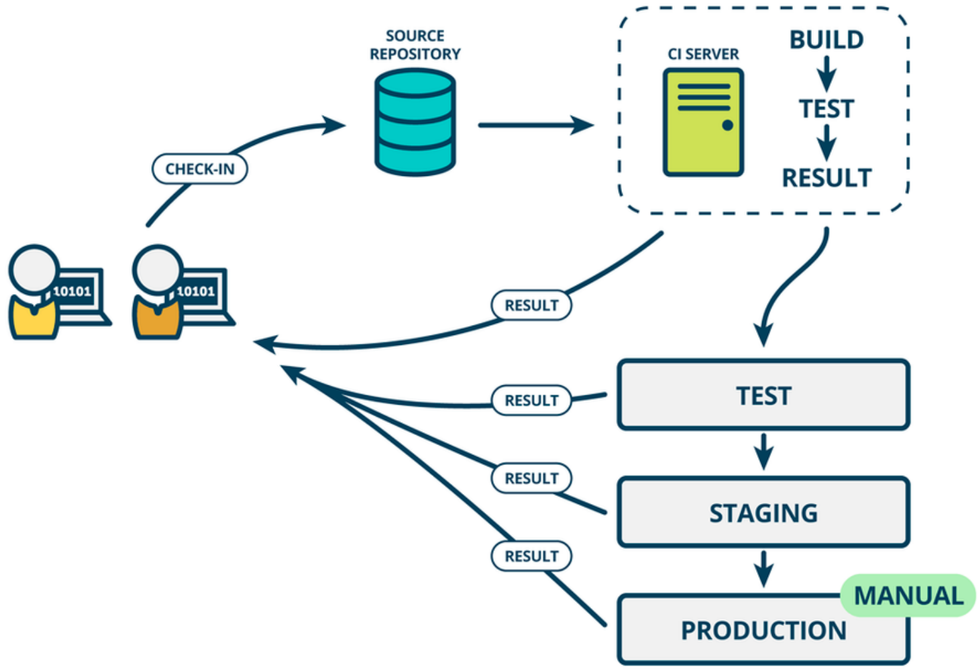
Is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.

What you need:

- Strong foundation in CI
- Enough tests
- Deploys need to be automated

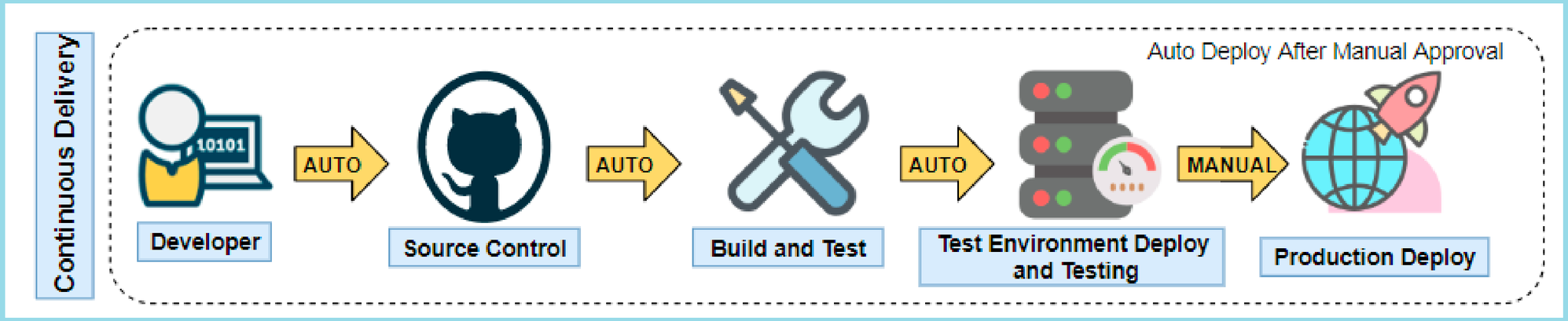
What you gain

- No more complexity of deploying software (not many days to release)
- Releases more often
- Much less pressure on decisions for small changes



Continuous Deployment

- One step further than continuous delivery this doesn’t need manual approval.
- Only a failed test will prevent a new change to be deployed to production
- Excellent way for feedback loop
- Rollback process

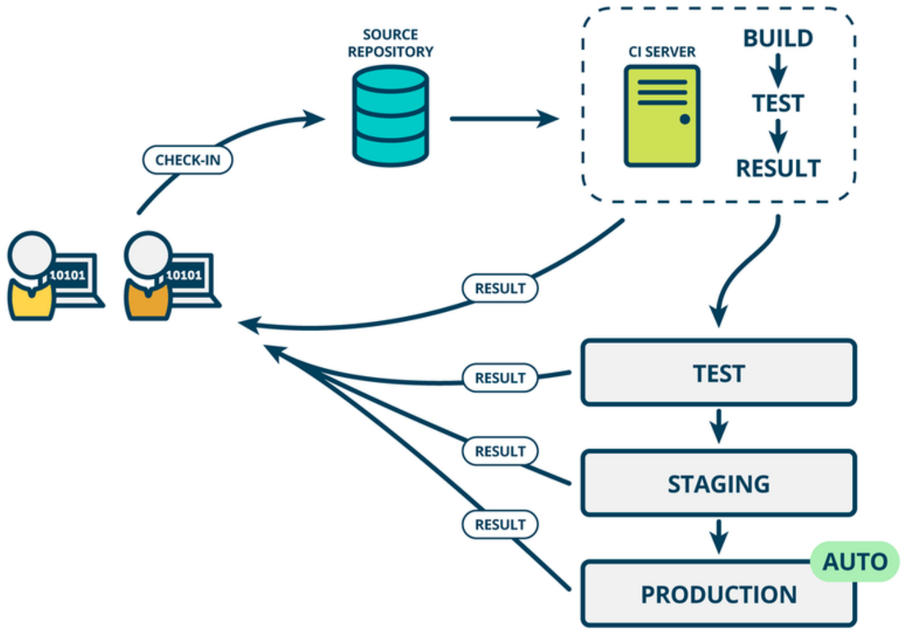


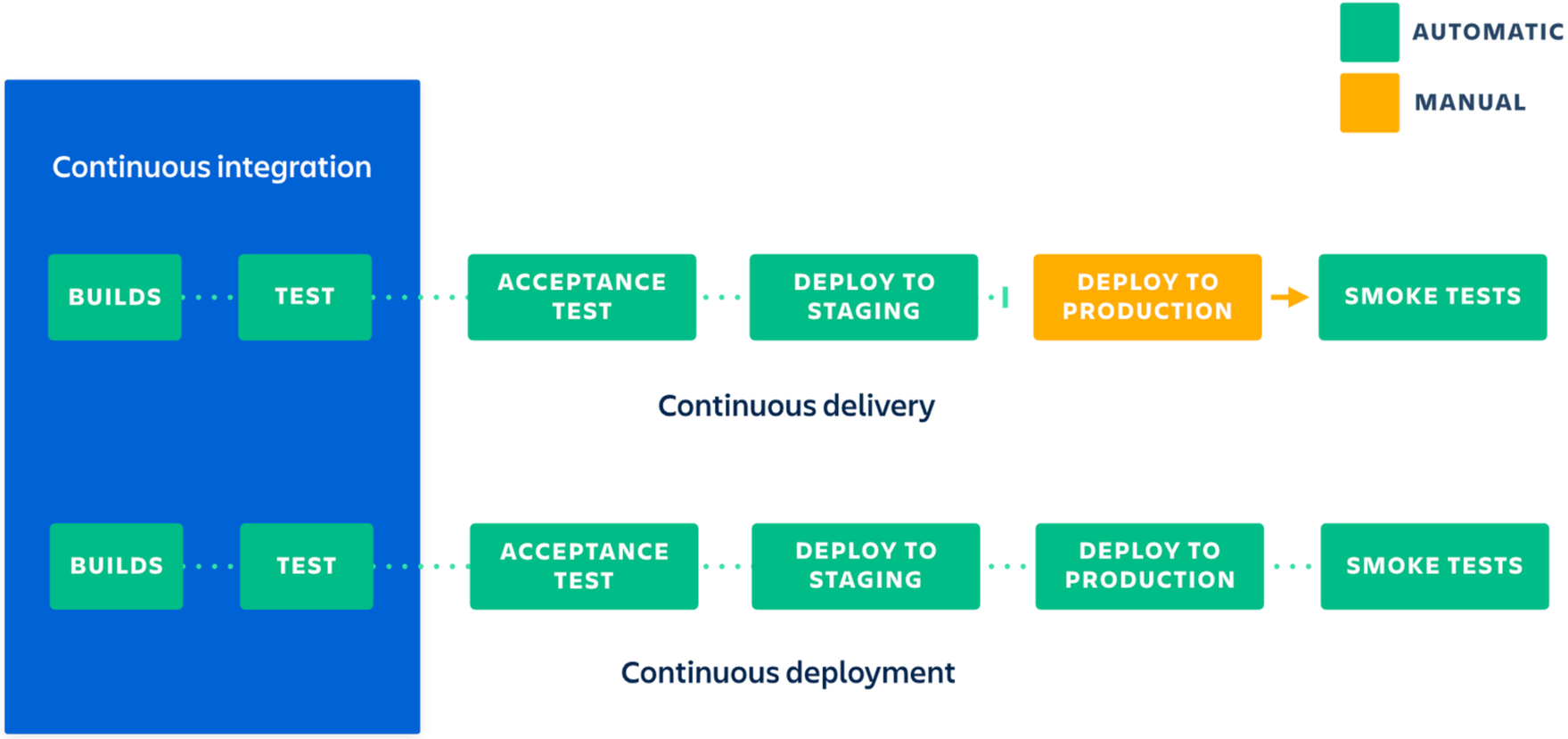
What you need:

- Documentation process of each step
- Coordination with other departments
- Have the best test culture and a good quality of tests***

What you gain

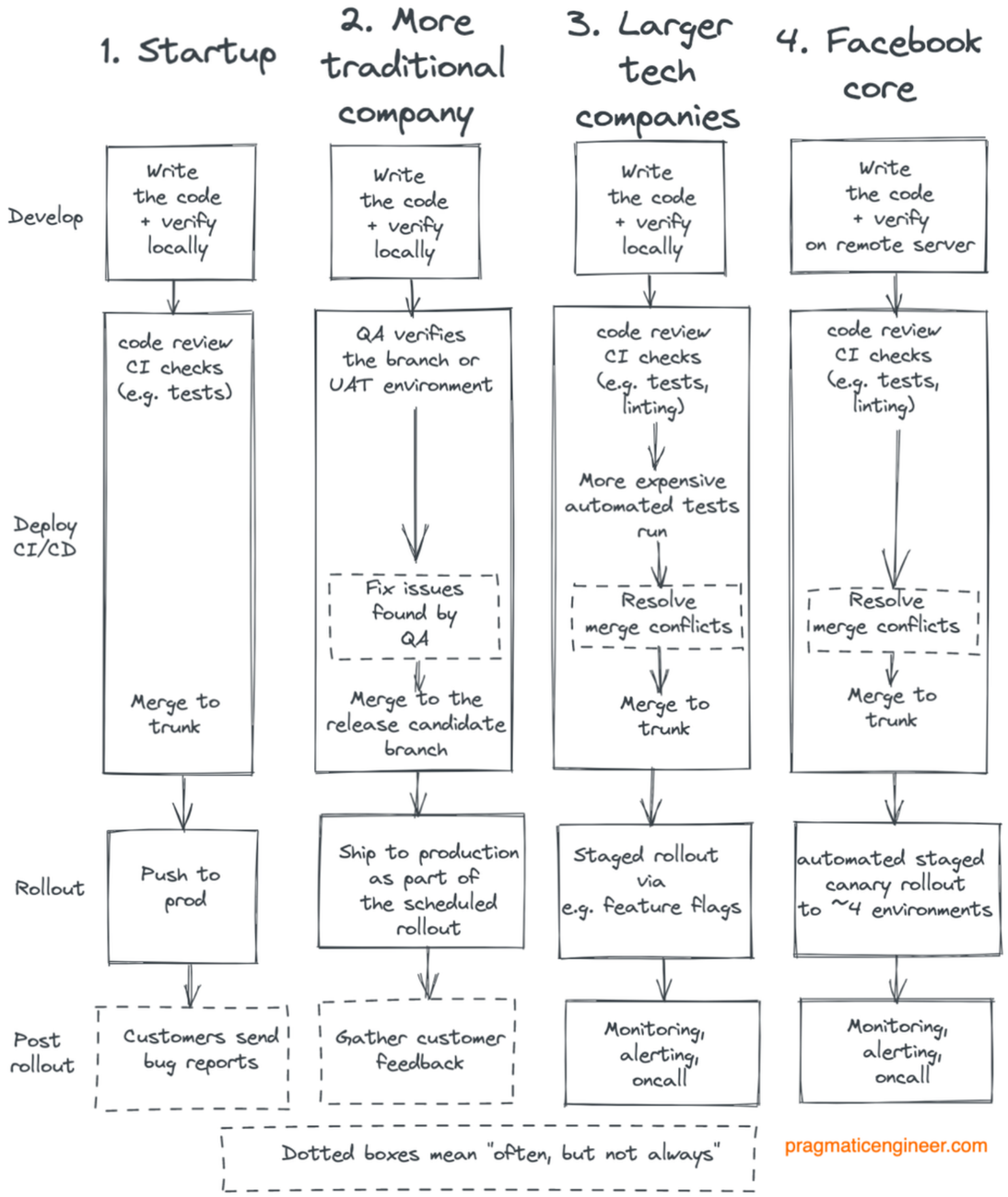
- develop process is faster, not necessity to pause development releases
- Deployments are automatically triggered
- Rollback process (auto)
- More improvements, faster.




AUTOMATIC
MANUAL
Continuous integration
BUILDS
TEST
ACCEPTANCE TEST
DEPLOY TO STAGING
DEPLOY TO PRODUCTION
SMOKE TESTS
Continuous delivery
BUILDS
TEST
ACCEPTANCE TEST
DEPLOY TO STAGING
DEPLOY TO PRODUCTION
SMOKE TESTS
Continuous deployment

How do various companies typically ship to production?

An admittedly imperfect attempt to visualize the differences in approaches



GFT

Author: Patrycjusz Czerniga

DevOps in the clouds
CZERNIGA.IT

Legend:

Tool type	›	Does the tool support the CI and CD process?
Free version	›	Is there a free version for the tool?
Pricing	›	Price for using the tool
Operating system	›	The operating system on which the tool can be installed.
Open source	›	Is the tool open source ?

Difficulty	›	Degree of difficulty in using the tool.
Plugins	›	Can the functions of the tool be extended by plugins?
Integration	›	Can tool's features be extended by integrations?
Platform	›	Can the tool be run on the cloud or on-premise?
Kubernetes support	›	Can the tool be installed on Kubernetes or can you run agents in on that platform?

External DB Needed	›	Do you need an external database to run the tool?
Built-in Git repository	›	Does the tool include a built-in Git repository?
Version control integration	›	Supported version control systems.
Plugin source	›	Source from which pluigny is taken
GitHub/Azure AD Authentication	›	Does the tool support Github/Azure AD authentication?

iOS/macOS support	›	Does the tool support iOS/macOS ?
Pipeline as a code	›	Can a tool define a pipeline as a code?
Container support	›	Does the tool support docker containers?
Best for	›	For which teams is a particular tool best?

	Tool type	Free version	Pricing	Operating system	Open source	Difficulty	Plugins	Integration	Platform	Kubernetes support	External DB Needed	Built-in Git repository	Version control integration	Plugin source	GitHub/Azure AD Authentication	iOS/macOS support	Pipeline as a code	Container support	Best for
Jenkins	CI/CD	✓	FREE	Windows, Linux, macOS	✓	Medium	●●●●●	●●●●●	On-premise & cloud	✓	✗	✗	GIT, Mercurial, TFS, SVN, Bazaar, CVS	Internal store	GitHub/Azure AD	✓	✓	✓	For teams that want to use the most widely used solution that provides the largest number of plugins and integrations.
GitLab	CI/CD	✓	0-99$ per user/month	Windows, Linux, macOS	✗	Medium	●●●●○	●●●●○	On-premise & cloud	✓	✗	✓	GIT	Internal store	GitHub/Azure AD	✓	✓	✓	For teams that want a comprehensive solution that includes a version control system and CI/CD/ tools
circleci	CI/CD	✓	0-2000$ per month	n/a	✗	Medium	●●●○○	●●●●●	Cloud	✓	n/a	✗	GIT	Internal store/ GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want a very fast build system with good GitHub integration.
TeamCity	CI/CD	✓	0-1500 euro per month	Windows, Linux, macOS	✗	Medium	●●●●○	●●●●○	On-premise & cloud	✓	Recomended: MySQL, MSSQL, PostgreSQL, Oracle	✗	GIT, Mercurial, Perforce, Subversion, Azure DevOps Server	Internal store/ GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want an easy-to-use solution that provides useful wizards to speed up configuration.
Bitbucket Pipelines	CI/CD	✓	0-6$ per user/month	n/a	✗	Medium	●●●○○	●●●○○	Cloud	✗	Recomended: MySQL, MSSQL, PostgreSQL, Oracle	✓	GIT, Subversion	Internal store	✗	✗	✓	✓	For teams that want a cloud solution that is fully integrated with another Atlassian tools (Jira and Bitbucket).
Buddy	CI/CD	✓	0-35$ per user/month	Linux, macOS	✗	Easy	●○○○○	●●●○○	On-premise & cloud	✗	✗	✗	GIT, AWS CodeCommit	n/a	GitHub	✓	✓	✓	For teams that want a very fast build system with easy setup.
Travis CI	CI/CD	✓ (free trial)	30-3300$ monthly	n/a	✗	Medium	●○○○○	●●○○○	Cloud	✗	n/a	✗	GIT	n/a	GitHub	✓	✓	✓	For teams that want a cloud system with good GitHub integration and nice interface.
CODESHIP	CI/CD	✓	0-23980$ per month	n/a	✗	Medium	●●●○○	●●○○○	Cloud	✗	n/a	✗	GIT	GitHub	✗	✗	✓	✓	For teams that want a solution that provides a set of tools for quickly creating and building our deployment artifacts and push them to the designated servers.
go	CI/CD	✓	FREE	Windows, Linux, macOS	✓	Medium	●●○○○	●●○○○	On-premise & cloud	✓	Recomended: MySQL, H2, PostgreSQL	✗	Git, Mercurial, SVN, TFS, Perforce	GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want a free and open source solution with parallel execution
	CI/CD	✓	Pay as you go	n/a	✗	Medium	●●○○○	●●○○○	Cloud	✗	n/a	✗	GIT	GitHub	GitHub	✓	✓	✓	For teams that want a very fast solution with simple configuration with Visual Pipeline Builder.
codemagic	CI/CD	✓	Pay as you go/ $299 per month	n/a	✗	Medium	●●●○○	●●●●○	Cloud	✗	n/a	✗	GIT	Internal store	GitHub/Azure AD	✓	✓	✓	For mobile development teams that want a building server for Flutter, React Native, Ionic and Cordova, as well as Native Android and Native iOS
Buildkite	CI/CD	✓	FREE	Windows, Linux, macOS	✓	Hard	●○○○○	●○○○○	On-premise	✓	Recomended: MySQL, PostgreSQL	✗	GIT, Mercurial, SVN, CVS, Baazar, Darcs	GitHub	GitHub	✓	✓	✓	For teams working in python (system and configuration is based on this language)
	CI/CD	✓	0-210$ per user/ month	Windows, Linux, macOS	✗	Medium	●●●●●	●●●●●	On-premise & cloud	✓	✗	✓	GIT, SVN	Internal store	GitHub/Azure AD	✓	✓	✓	For teams that are already using the GitHub solution.
Concourse	CI/CD	✓	FREE	Linux, macOS	✓	Medium	●●●○○	●●●○○	On-premise	✓	PostgreSQL	✗	GIT, SVN, Mercurial	GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want a quick tool where everything can be defined in code: tasks, server configuration, worker. This makes it easy to update or move the solution.
TEKTON	CI/CD	✓	FREE	Windows, Linux, macOS	✓	Medium	●●●●○	●●○○○	On-premise & cloud	✓	✗	✗	GIT	Internal store	✗	✓	✓	✓	For teams that want a cloud-native solution for building CI/CD pipelines. Tekton installs and runs as an extension on a Kubernetes cluster and comprises a set of Kubernetes Custom Resources that define the building blocks you can create and reuse for your pipelines.
Bamboo	CI	✓ (free trial)	1200-187380$	Windows, Linux, macOS	✗	Hard	●●●●○	●●●●●	On-premise	✗	Recomended: MySQL, MSSQL, PostgreSQL, Oracle	✗	GIT, Mercurial, Perforce, Subversion, CVS	Internal store/ GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want a on-premise solution that is fully integrated with another Atlassian tools (Jira and Bitbucket).
DRONE by harness	CI	✓	0 - $299 per month	Linux, macOS	✓	Medium	●●●●○	●●●●○	On-premise & cloud	✓ (beta)	Recomended: MySQL, PostgreSQL	✗	Git, Mercurial, Bazaar, SVN	Internal store	GitHub	✓	✓	✓	For teams that want to run their tasks in containers in a local environment.
harness	CD	✓	0-100$ per service/ month	Linux, macOS	✓	Medium	●●●●○	●●●●○	On-premise & cloud	✓	MongoDB	✗	GIT, CodeCommit, Azure DevOps	Internal store	GitHub/Azure AD	✓	✓	✓	For teams that want to use a simple CD tool that gives interesting features such as: feature flags, infrastructure-as-code, cloud costs, change tracking.
argo	CD	✓	FREE	Windows, Linux, macOS	✓	Medium	●●○○○	●●○○○	On-premise & cloud	✓	✗	✗	GIT	GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want a cloud-native continuous deployment (CD) processes. AgroCD use Git as the source of truth and is Enterprise-friendly.
flux	CD	✓	FREE	Windows, Linux, macOS	✓	Medium	●●○○○	●●●○○	On-premise & cloud	✓	✗	✗	GIT, AWS CodeCommit, Azure DevOps	GitHub	✗	✓	✓	✓	For teams that want a solution that deploy apps with canaries, feature flags, and A/B rollouts. Flux can also manage any Kubernetes resource. Infrastructure and workload dependency management is built in.
Spinnaker	CD	✓	FREE	Linux, macOS	✓	Medium	●●○○○	●●○○○	On-premise & cloud	✓	✗	✗	GIT	GitHub	GitHub/Azure AD	✓	✓	✓	For teams that want a delivery server for releasing software changes at a very high speed.
AWS CodePipeline	CI/CD	✓	Pay as you go	n/a	✗	Medium	●●●○○	●●●○○	Cloud	✗	n/a	✓	GIT	Internal store	✗	✓	✓	✓	For teams that are already using the AWS Cloud.
Azure Pipelines	CI/CD	✓	Pay as you go	n/a	✗	Medium	●●●●○	●●●●○	Cloud	✓	n/a	✓	GIT	Internal store	Azure AD	✓	✓	✓	For teams that are already using the Azure Cloud.
	CI/CD	✓	Pay as you go	n/a	✗	Medium	●●●○○	●●●○○	Cloud	✗	n/a	✓	GIT	GitHub	✗	✗	✓	✓	For teams that are already using the Google Cloud.

Last Updated: 03.04.2022

https://www.czerniga.it/2022/03/27/find-your-best-ci-cd-tool/
Tomasz Gintowt

Strategies of deployment

Recreate

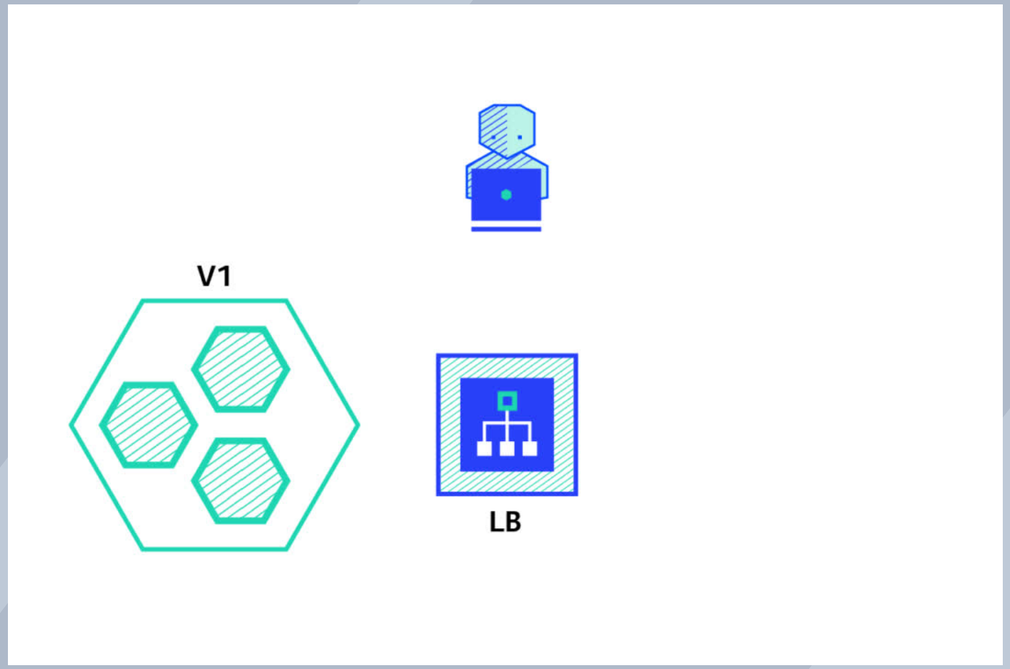
Version A is terminated then version B is rolled out.

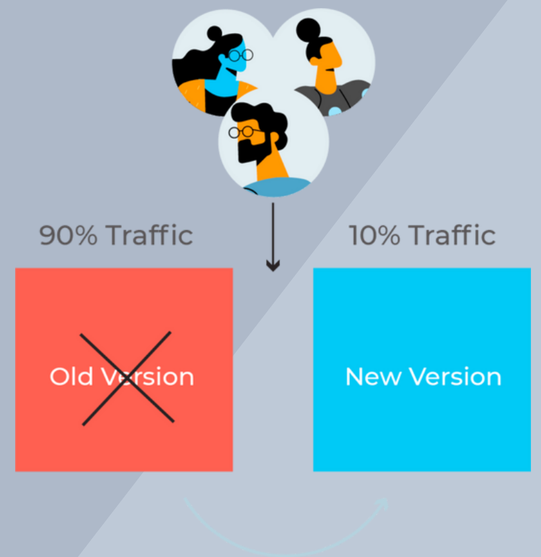
Pros:

- Easy to setup
- App completely renewed and fresh
- Deal with one scaling process at time

Cons:

- High impact on the user
- Downtime
- Time depends of the time you end one app and start the other one





Ramped

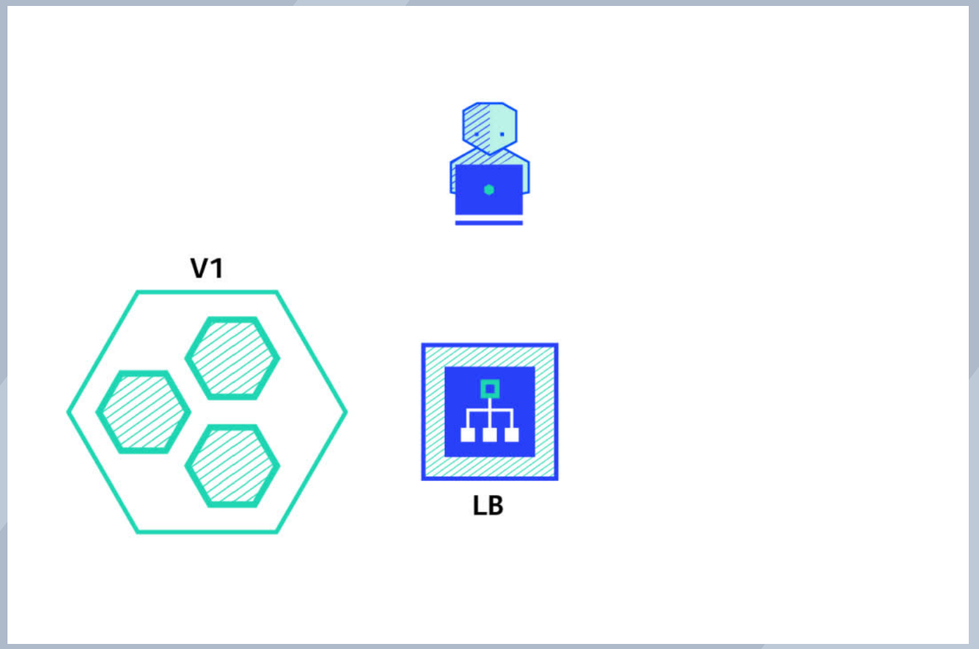
Slow rolling out of a version and replacing instances one after other.

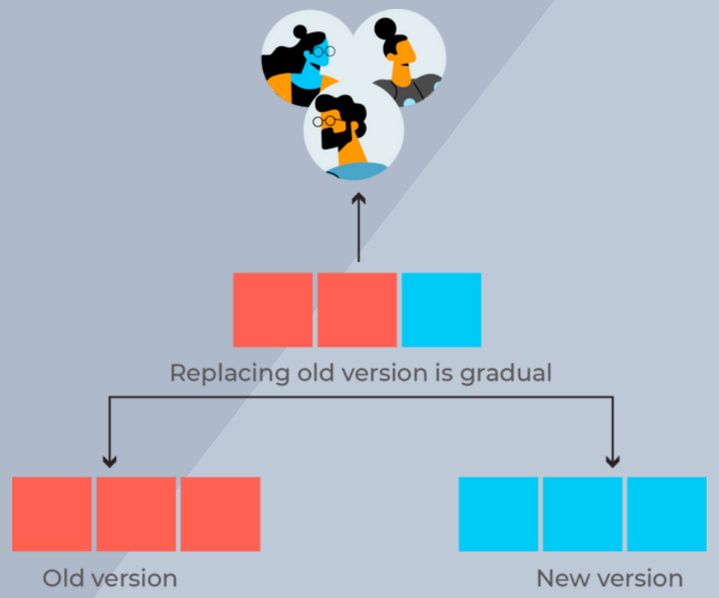
Pros:

- Easy to setup
- If you have a load balancer this option fits you fine
- Zero downtime
- Enable Performance monitoring

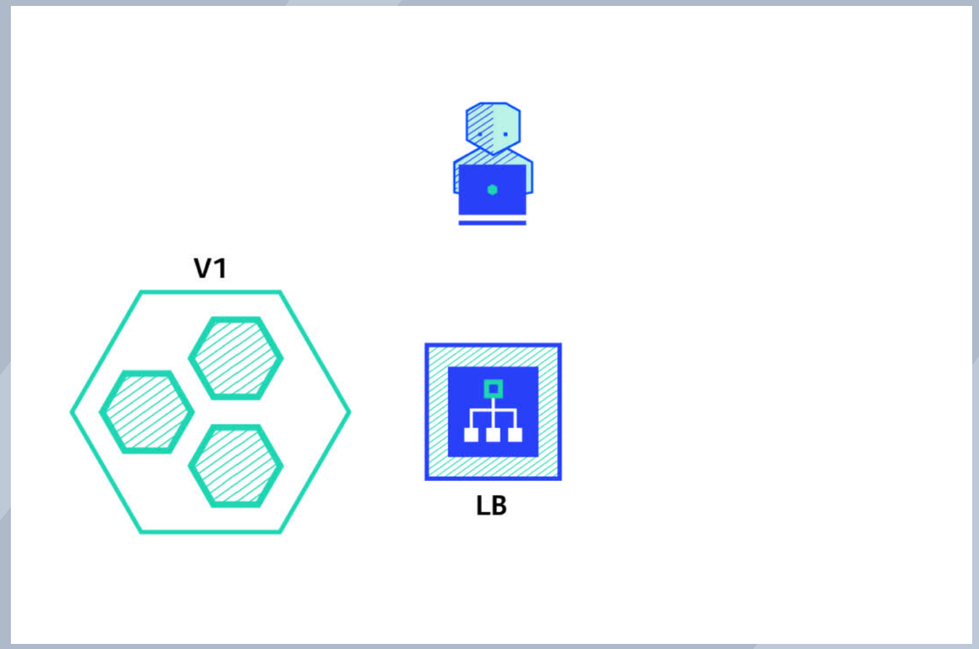
Cons:

- Rollback is difficult (time)
- No much control on traffic
- Not recommended for multiple APIs because is difficult to manage and rollback if needed





Blue / Green



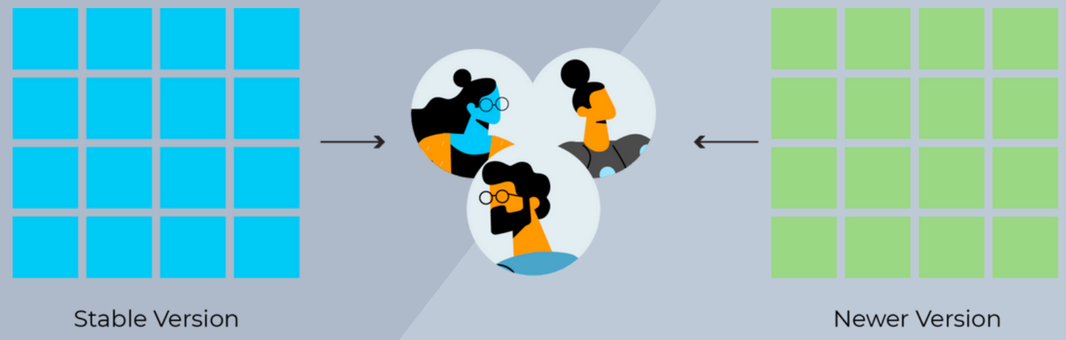
You create a complete new version of your system and once this new env is ready, you redirect your traffic into it.

Pros:

- Instant rollout or rollback if needed
- No problem to have many versions, all new version is realized
- You don’t switch the traffic until you are sure is ready

Cons:

- Expensive (double resources)
- Test is needed before to release the new version
- Stateful applications can be hard



Canary

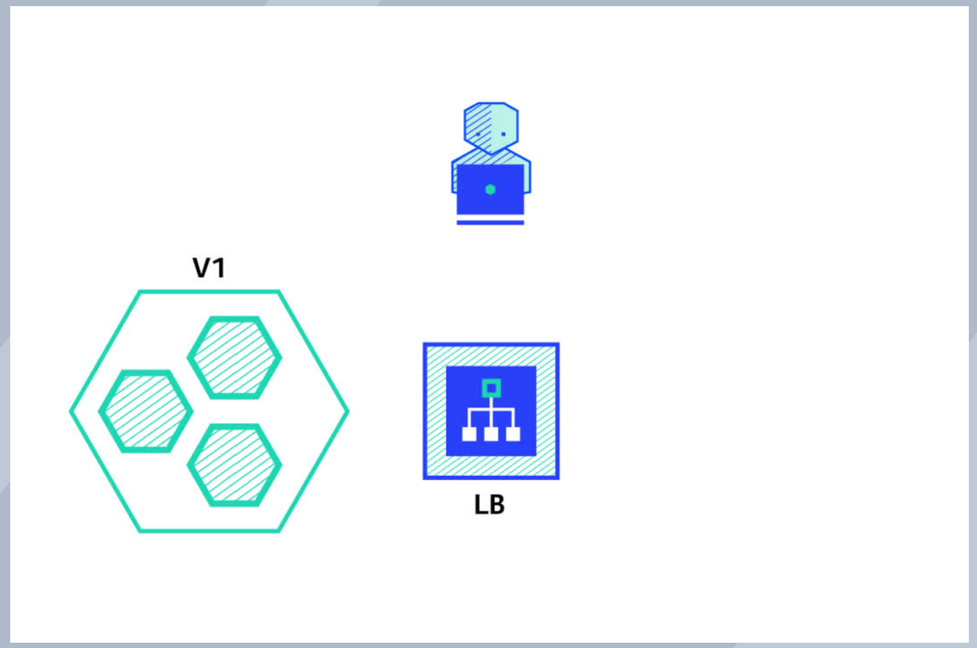
Gradually shifting prod traffic from older to the newer version

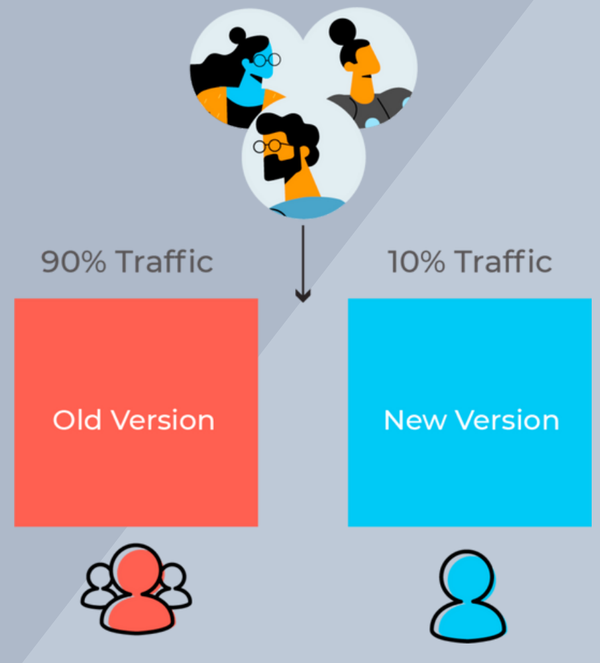
Pros:

- Version is released for a subset of users
- Fast rollback
- Convenient for performance monitoring

Cons:

- Slow roll out and more time time consuming





A/B Testing

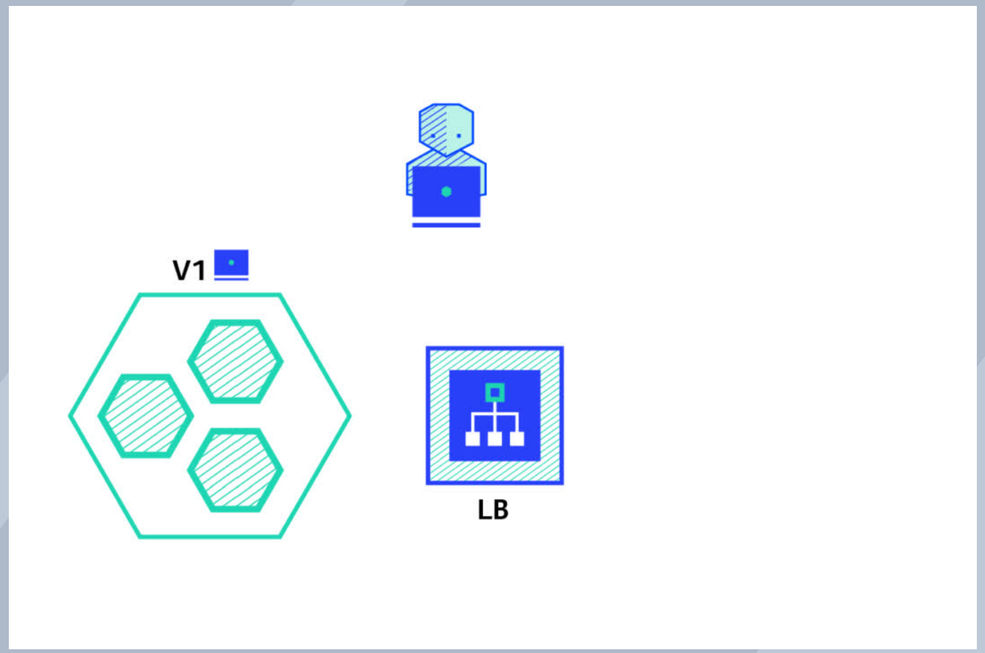
Route a partial subnet of users to a new functionality

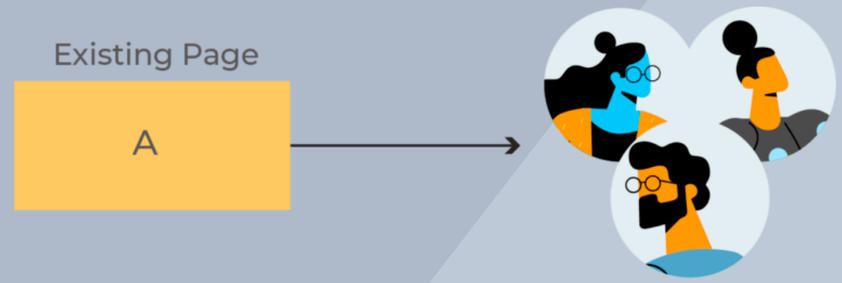
Pros:

- Several versions running in parallel
- Full control over the traffic
- Fast feedback
- Measure the effectiveness of the new version

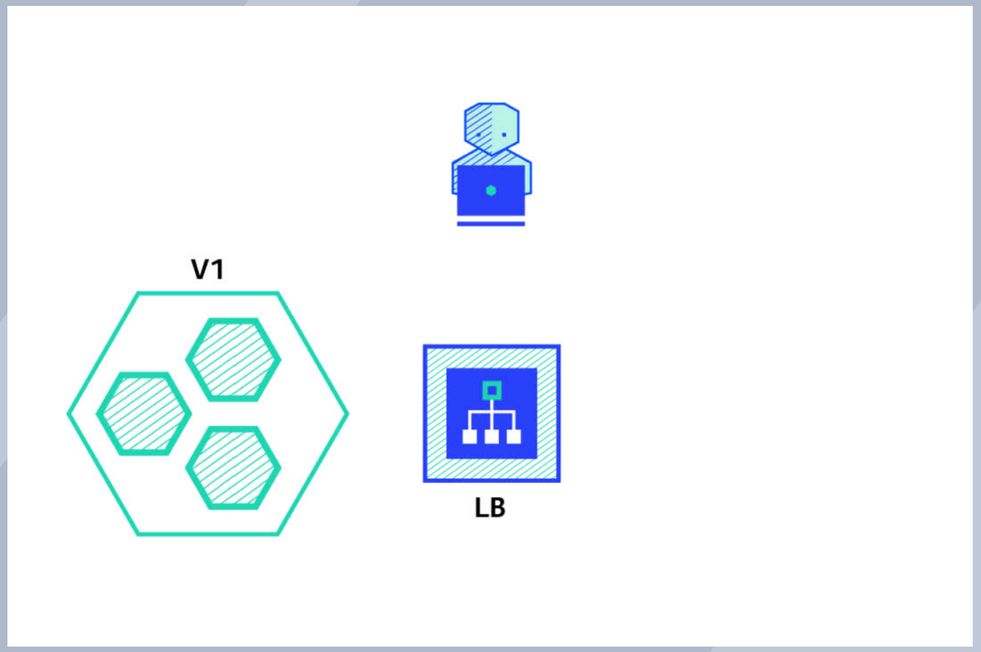
Cons:

- Requiere a good load balancer
- Hard to troubleshoot for a given sesion





Shadow



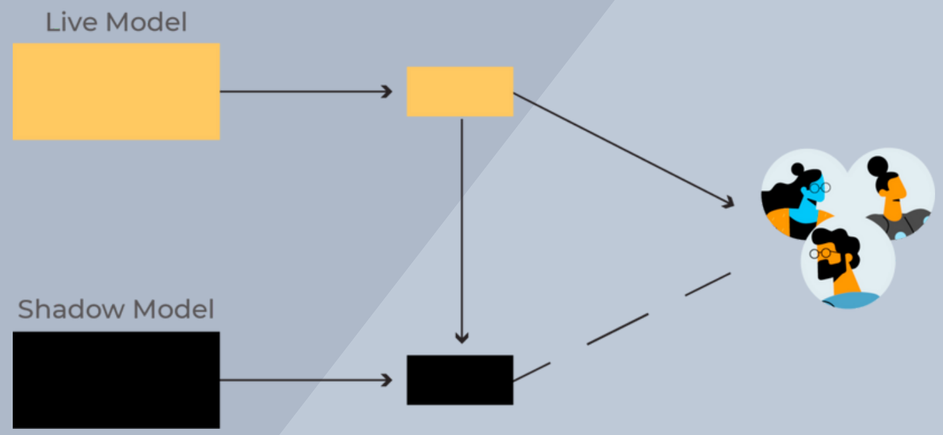
Deploy a new version along to the old one. But users don’t access to it immediately

Pros:

- Real traffic for testing
- Not impact on the user
- No rollout until the performance meet requirements

Cons:

- Double resources
- Need to simulate a response to the user to avoid duplication of requests
- Complex to setup



Questions

References

- https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment
- https://geekflare.com/es/understanding-ci-cd/
- https://www.youtube.com/watch?v=HnWuIjUw_Q8
- https://www.cloudbees.com/continuous-delivery/continuous-integration
- https://thenewstack.io/deployment-strategies/
- https://www.plutora.com/blog/deployment-strategies-6-explained-in-depth


THANK YOU