Christos Galanopoulos

DevOps Engineer @ Performance Technologies

Hands-on experience architecting, automating, and optimizing mission-critical deployments across complex infrastructures. Proficient with CI/CD pipelines, "as code" tools, cloud operations, container orchestration, and scripting.

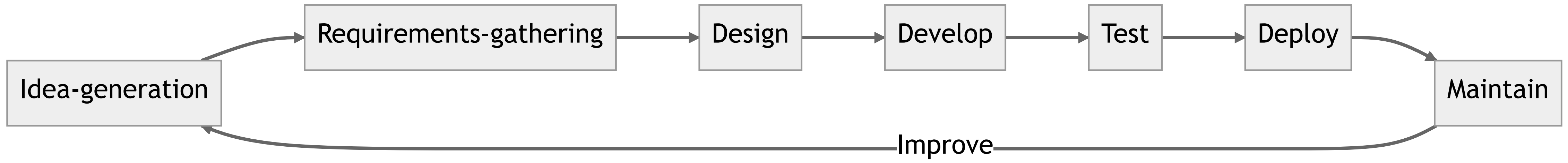
Going with the dev-flow

Why and how should we embrace and integrate the dev-flow into cloud operations?

Definition

dev-flow, short for development flow, refers to the process of software development from ideation to deployment. It involves various stages that aim to ensure the production of quality software products.

Typical flow



By following a standardized process, teams can collaborate effectively, improve efficiency, and ensure the delivery of high-quality products and services.

Our main focus

- **Design**
  - infrastructure
  - governance
- **Develop**
  - infrastructure as code
  - configuration as code
  - policy as code

- **Test**
  - validate code modules
  - introduce what-if checks
  - include tests (unit, integration etc.)
- **Deploy**
  - adopt a standard process
  - review changes
  - include branch protection rules

Three pillars

- "as code" mindset
- automation
- process standardization

Why adopt an "as code" mindset?

- faster provisioning
- consistency
- scalability
- reusability
- version control
- improved security

Version control

Version control makes it possible to track changes, collaborate on code, and revert to previous versions if necessary. It improves quality assurance, collaboration, transparency, and organization while lowering the risk of data loss.

Automation

Automation can be implemented in almost all phases of a project. From the development and testing of code to the actual deployment. Pipelines can be created on a variety of platforms to automate repetitive tasks so that they execute quickly, consistently, and reliably.

Process standardization

It is critical to have a standardized process when working on a project in the manner described.

For example, how you organize your code in a repository, which branching strategy you employ, code discussions, reviews that must take place prior to merging changes, and so on.

You **should not reinvent the wheel** with each new project; instead, you should have a standard practice that you follow and may slightly deviate from.

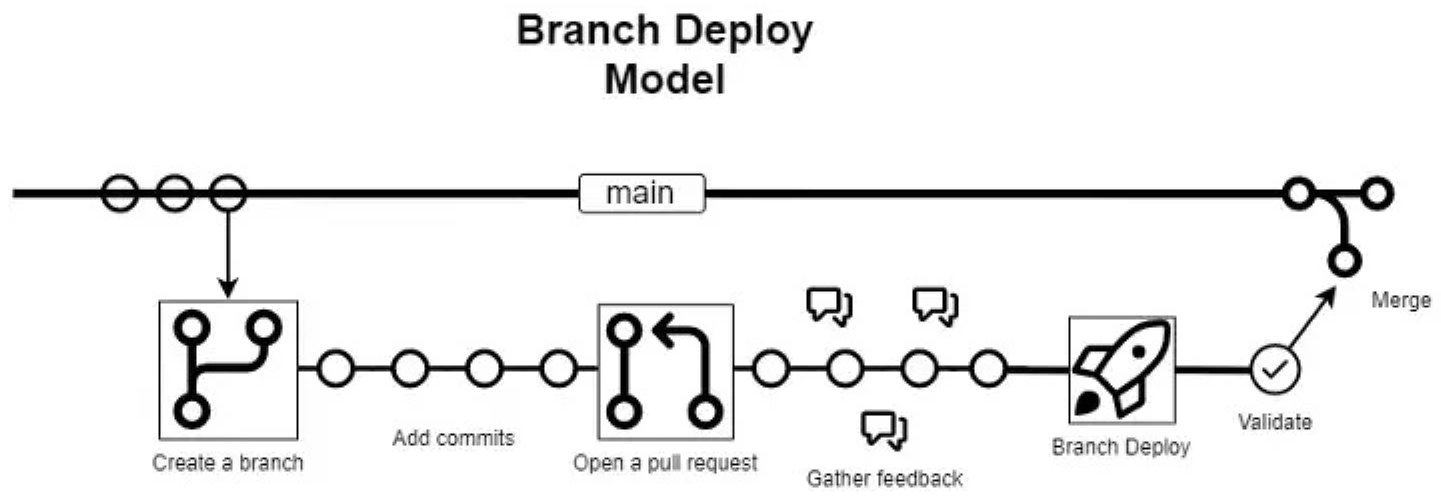
Tools

- **Version control**: Git, SVN, ...
- **Code hosting**: GitHub, GitLab, ...
- **Code tools**: Terraform, Bicep, Ansible, ...
- **CI/CD**: GitHub Actions, Jenkins, GitLab, ...
- **Testing**: Terratest, PSRule, Checkov, ...

Our goal today

Deploy a simple API written in Python to an Azure WebApp using GitHub Actions, Bicep, and PSRule.

Branch deploy



Benefits

- Improved efficiency
- Simpler debugging and testing
- Early feedback
- Faster cycle time
- Consistent results
- Reduced costs
- CI/CD capabilities
- Efficient collaboration
- Improved risk management

Things to keep in mind

- "as code" mindset
- version control
- everything can and should be tested
- minimized human factor in error prone tasks
- automation, automation, automation ...
- strive for confident deployments
- process standardization
- meaningful collaboration

Personal information

- **Blog**: https://christosgalano.github.io/
- **Email**: christosgalanop@gmail.com
- **dev.to**: https://dev.to/christosgalano
- **GitHub**: https://github.com/christosgalano
- **LinkedIn**:https://www.linkedin.com/in/christos-galanopoulos/