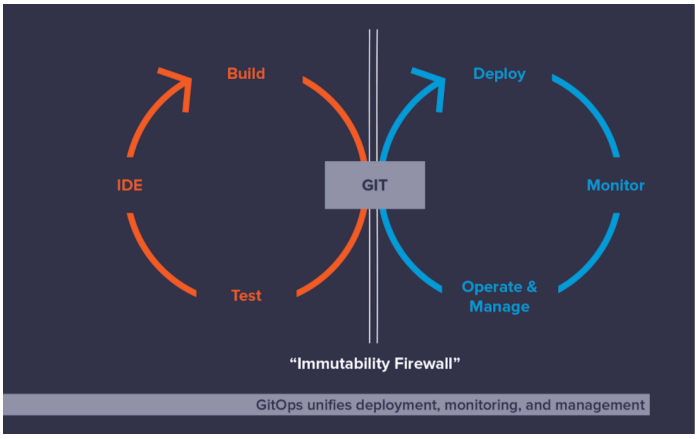
GitOps principles



#1. Your entire system described declaratively

Kubernetes is one of many modern cloud native tools out there that is managed with declarative configurations. Declarative means that configuration is guaranteed by a set of facts instead of by a set of instructions. This means that configuration can be treated as code and kept in Git alongside your application source code. But more importantly, with your entire system's declarative configuration under source control, you have a single source of truth of your system's desired state, providing a number of benefits like the ability to simultaneously manage infrastructure and application code.

#2. A desired system state version controlled in Git

With the declarative definitions of your system stored in Git, and serving as your canonical source of truth, there is a single place from which your cluster can be managed. This also trivializes rollbacks and roll forwards to take you back to a 'good state' if needed. With Git's excellent security guarantees, SSH key signed commits to enforce strong security guarantees about authorship as well as the code's provenance.

#3. The ability for changes to be automatically applied

Once the desired system state is kept in Git, the next step is the ability to automatically reconcile changes with your system. What's significant about this is that you don't need specific cluster credentials to make a change. With GitOps, there is a separated environment that the state definition lives outside of. This allows your team to separate what they actually do, for example, deploying code from how they are going to do it, for example, configuring the cluster to deploy the code.

#4. Software agents that verify correct system state and alert on divergence

With the desired state of your entire system kept under version control, and running in the cluster, you can now employ software controllers to bring the actual state of the cluster in alignment with that desired state, and inform you whenever reality doesn't match your expectations. The use of a GitOps controller like Flux in combination with these software agents ensures that your entire

system is self-healing. And by self-healing, we don't mean when nodes or pods fail, those are handled by Kubernetes, but in a broader sense, in the case of human error, for example. In this case, software agents act as the feedback and control loop for your operations.

Key GitOps benefits

Stronger security guarantees

Git's strong correctness and security guarantees, backed by Git's strong cryptography used to track and manage changes, as well as the ability to SSH sign changes to prove authorship and origin are key to a correct and secure definition of the cluster's desired state. If a security breach does occur, the immutable and auditable source of truth can be used to recreate a new system independently of the compromised one, reducing downtime and allowing for a much better incident response and more effective disaster recovery to meet compliance.

Also, the separation of responsibility between integrating and testing software, then releasing it to a production environment embodies the security principle of least privilege, reducing the impact of compromise and providing a smaller attack surface.

Increased speed and productivity

continuous deployment automation with an integrated feedback and control loop speeds up your mean time to deployment by supporting more frequent releases. Declarative definitions kept in Git enable developers to use familiar workflows, reducing the time it takes to spin up new development or test environments to deploy new features. Your teams can ship more changes per day and this translates into faster turnaround for new features and functionality to the customer.

Reduced mean time to detect and mean time to recovery

the amount of time it takes to recover from a cluster meltdown is also decreased with GitOps best practices. With Git's built in capability to revert/rollback and fork, you gain stable and reproducible rollbacks. Since your entire system is described in Git, you have a single source of truth for what to recover after a cluster failure, reducing your meantime to recovery (MTTR) from hours or days to minutes. GitOps provides real time feedback and control loops. In conjunction with other tools like Prometheus for observability and Jaeger for end-to-end tracing, problems can be detected and tracked down, preventing entire cluster meltdowns more quickly, and overall reducing mean time to detect (MTTD) and mean time to locate (MTTL).

Improved stability and reliability

due to GitOps providing a single operating model for making infrastructure and apps, you have consistent end-to-end workflows across your entire organization. Not only are your continuous integration and continuous deployment pipelines all driven by pull requests, but your operations tasks are also fully reproducible through Git.

Easier compliance and auditing

by incorporating Git as part of your cluster management strategy, you automatically gain a convenient audit trail of who did what and when for all cluster changes outside of Kubernetes that can be used to meet SOC 2 compliance and also ensure stability.