UbuCon India 25

GitOps + Ubuntu: Managing Infrastructure, Snap & Cloud Deployments Declaratively

Qualcomm Canonical IT'S FOSS

Neel Shah - DevOps Community Guy

- Developer Advocate at Middleware
- Hashicorp Ambassador
- Co-organiser GDG Cloud Gandhinagar, CNCF and Hashicorp Gandhinagar
- Mentored more than 15+ hackathons
- Gave talks in 10+ conferences, including KubeCon,Platform Con, LinuxFest, KCD, etc.

The GitOps Mindset

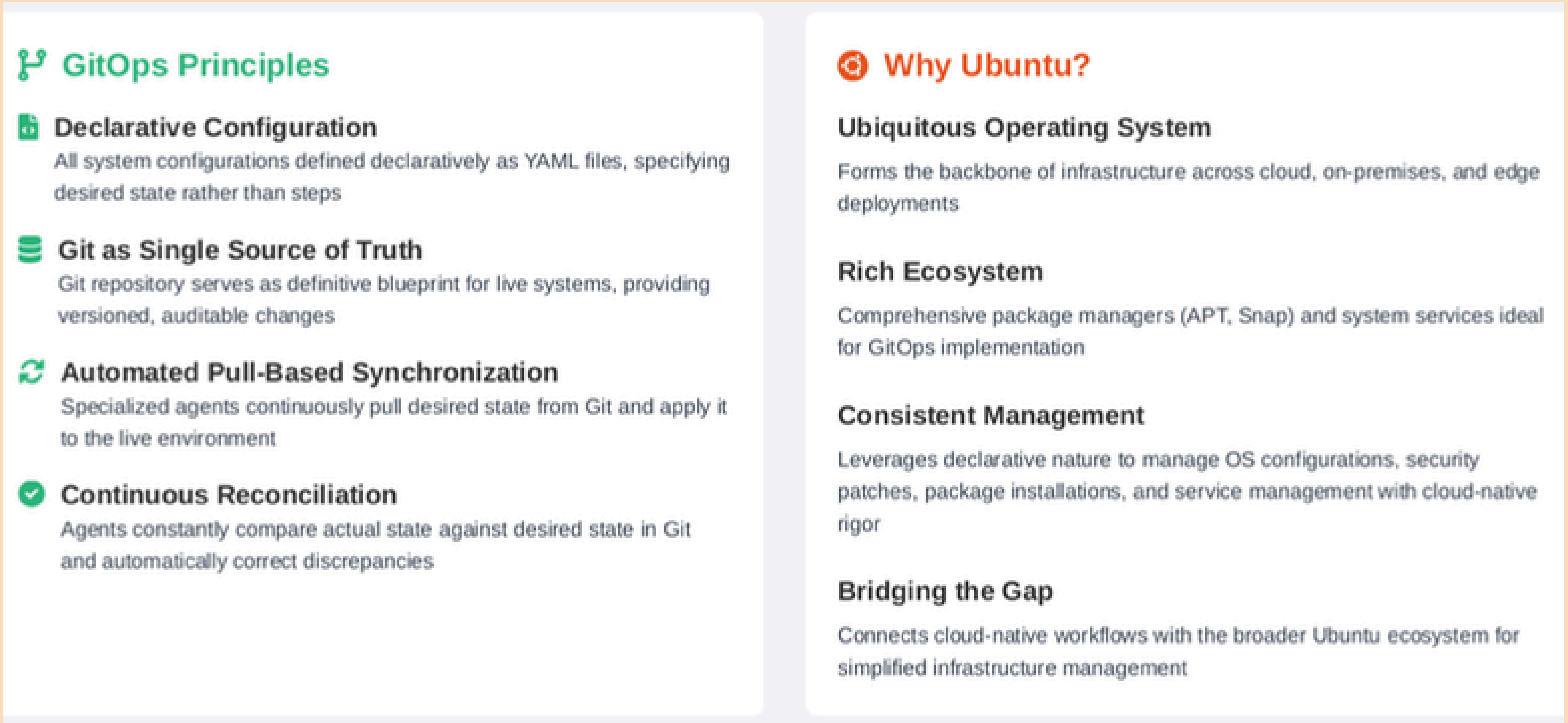
Git is the single source of truth

Desired state is stored in version control

Automation tools reconcile live systems with Git

Rollbacks, drift detection, and traceability baked in

Gitops beyod Kubernetes



GitOps Principles

Declarative Configuration
All system configurations defined declaratively as YAML files, specifying desired state rather than steps

Git as Single Source of Truth
Git repository serves as definitive blueprint for live systems, providing versioned, auditable changes

Automated Pull-Based Synchronization
Specialized agents continuously pull desired state from Git and apply it to the live environment

Continuous Reconciliation
Agents constantly compare actual state against desired state in Git and automatically correct discrepancies

Why Ubuntu?

Ubiquitous Operating System
Forms the backbone of infrastructure across cloud, on-premises, and edge deployments

Rich Ecosystem
Comprehensive package managers (APT, Snap) and system services ideal for GitOps implementation

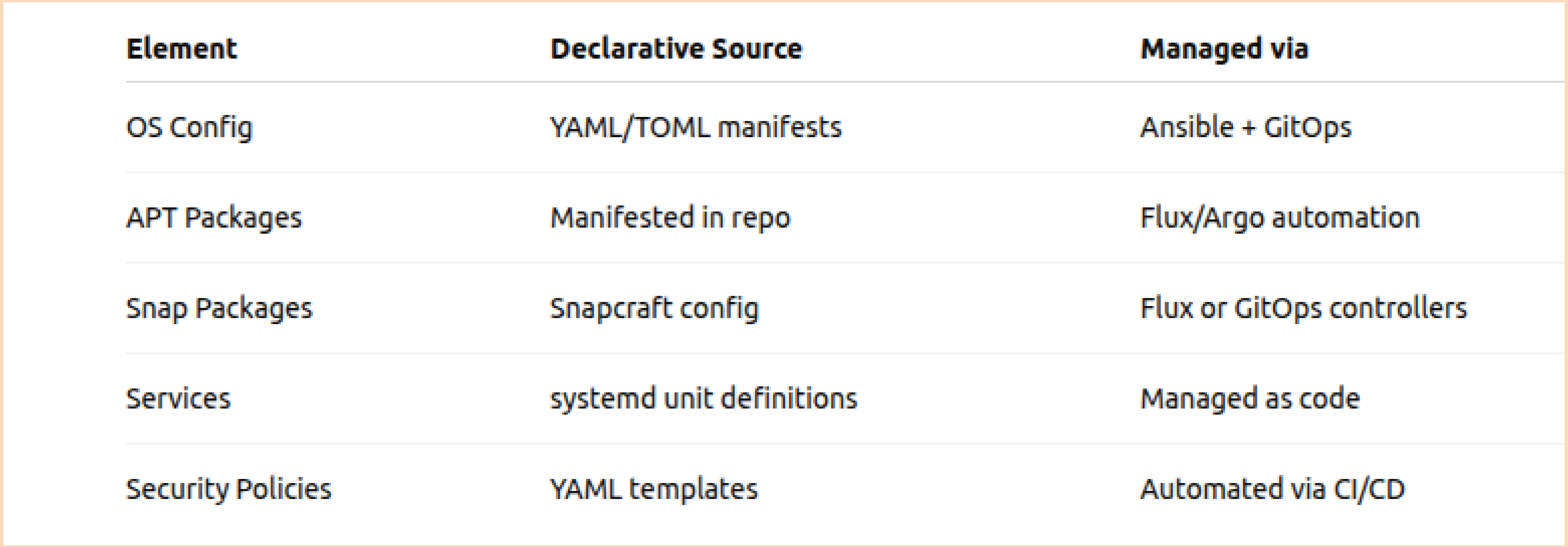
Consistent Management
Leverages declarative nature to manage OS configurations, security patches, package installations, and service management with cloud-native rigor

Bridging the Gap
Connects cloud-native workflows with the broader Ubuntu ecosystem for simplified infrastructure management

Why Extend GitOps to Ubuntu?

- Ubuntu powers cloud, desktop, and IoT/edge devices
- Traditionally configured imperatively (commands, scripts)
- Increasing need for consistency and auditable automation
- Bridge between DevOps, Platform Engineering, and InfraOps

Declarative Ubuntu Management



Element	Declarative Source	Managed via
OS Config	YAML/TOML manifests	Ansible + GitOps
APT Packages	Manifested in repo	Flux/Argo automation
Snap Packages	Snapcraft config	Flux or GitOps controllers
Services	systemd unit definitions	Managed as code
Security Policies	YAML templates	Automated via CI/CD

Tools of the Trade — Flux, ArgoCD, and Friends

- Flux for lightweight GitOps sync and reconciliation
- ArgoCD for enterprise-scale orchestration
- Ansible, Juju, or Terraform for provisioning layers
- Webhooks or GitHub Actions for change automation

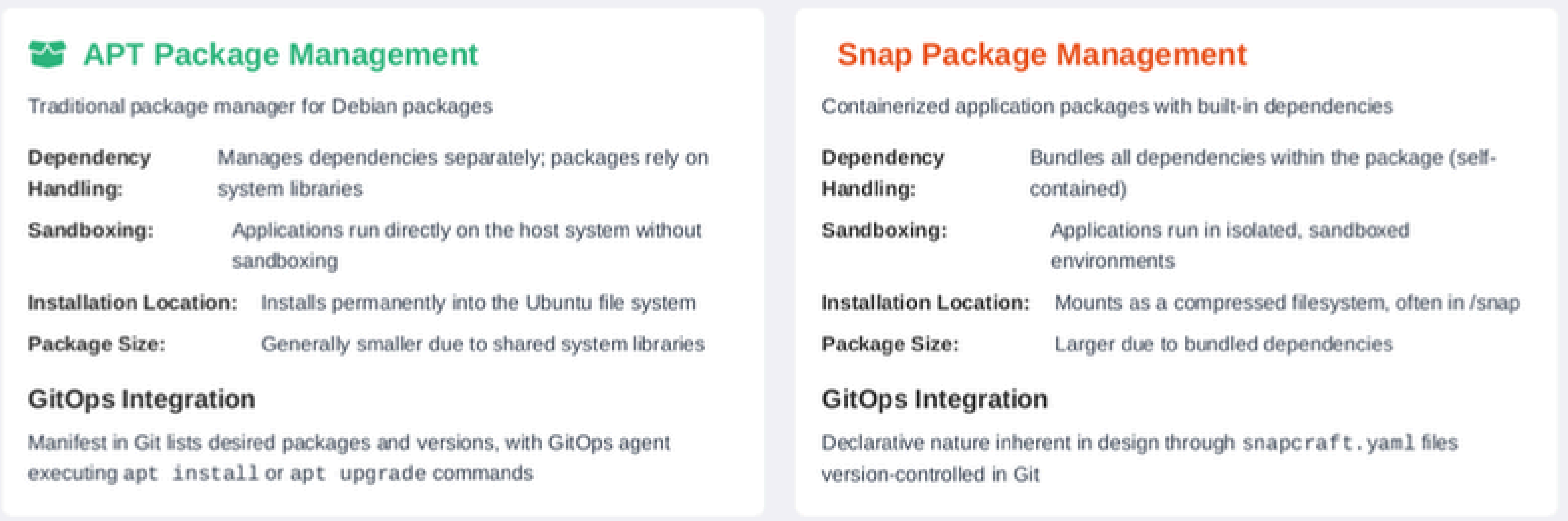
Managing Cloud, Edge, and On-Demand Ubuntu

01 Cloud: Provision and patch EC2/GCE/Azure VMs declaratively

02 Edge: Apply Snap updates and security patches from Git

03 On-Demand: Use Git commits to trigger Terraform + OS updates

04 Common GitOps workflow regardless of location



APT Package Management

Traditional package manager for Debian packages

Dependency Handling:	Manages dependencies separately; packages rely on system libraries
Sandboxing:	Applications run directly on the host system without sandboxing
Installation Location:	Installs permanently into the Ubuntu file system
Package Size:	Generally smaller due to shared system libraries

GitOps Integration

Manifest in Git lists desired packages and versions, with GitOps agent executing `apt install` or `apt upgrade` commands

Snap Package Management

Containerized application packages with built-in dependencies

Dependency Handling:	Bundles all dependencies within the package (self-contained)
Sandboxing:	Applications run in isolated, sandboxed environments
Installation Location:	Mounts as a compressed filesystem, often in /snap
Package Size:	Larger due to bundled dependencies

GitOps Integration

Declarative nature inherent in design through `snapcraft.yaml` files version-controlled in Git

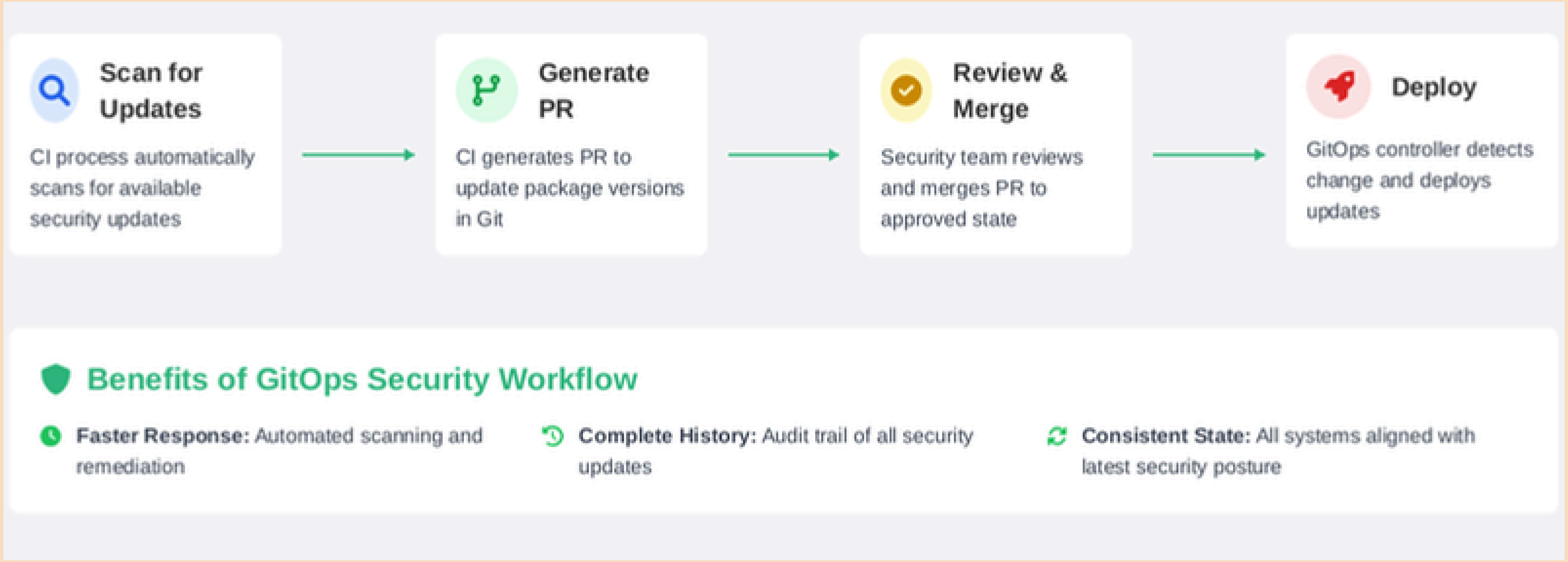
APT and Snap Package Management

Security, Patching, and Compliance

→

- ✓ OS security updates triggered by merge commits
- ✓ Drift detection for unauthorized changes
- ✓ Automatic rollback to last known good config
- ✓ Auditable and compliant change history

Automated Security Patching



Scan for Updates

CI process automatically scans for available security updates

Generate PR

CI generates PR to update package versions in Git

Review & Merge

Security team reviews and merges PR to approved state

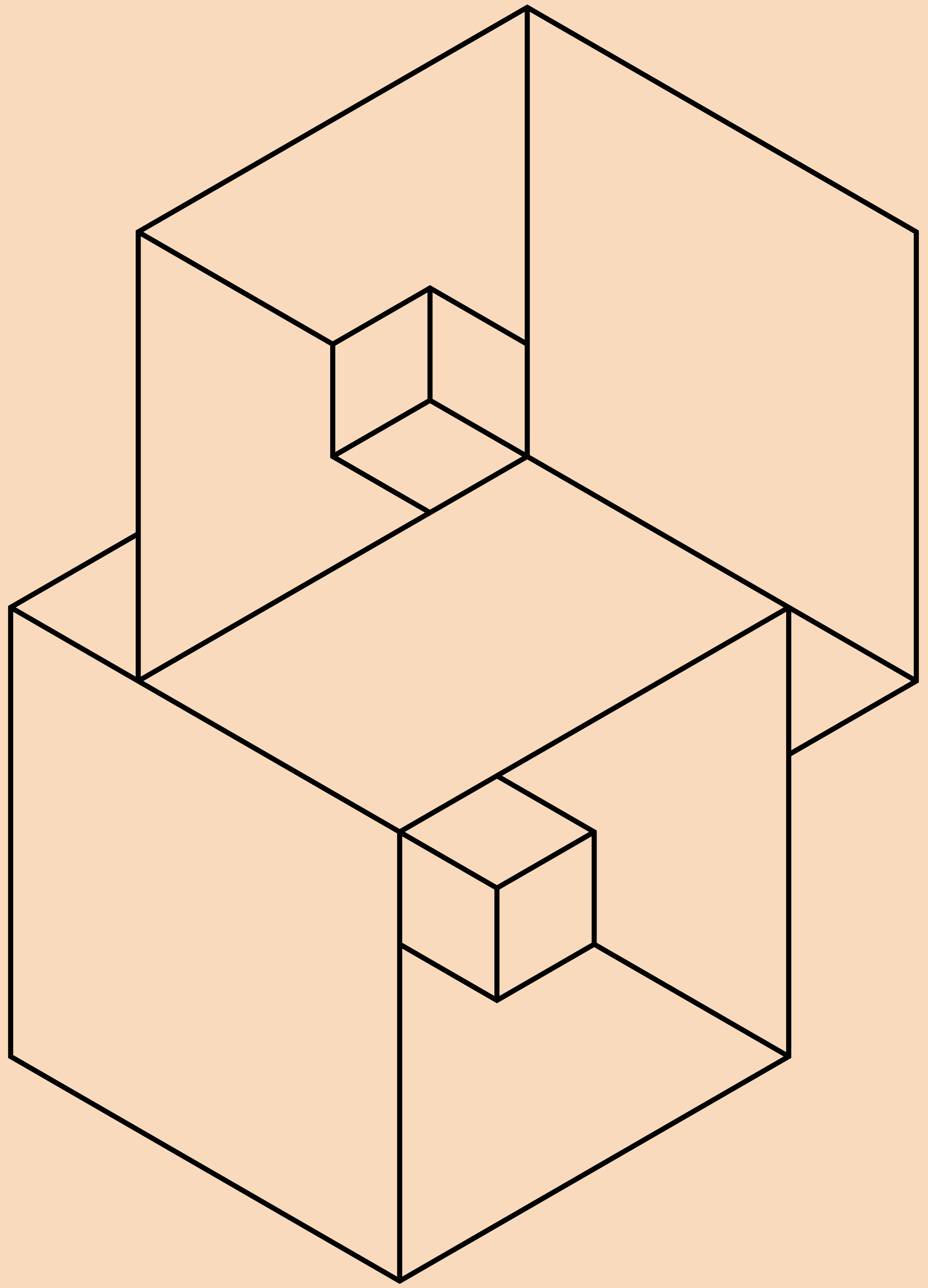
Deploy

GitOps controller detects change and deploys updates

Benefits of GitOps Security Workflow

- **Faster Response:** Automated scanning and remediation
- **Complete History:** Audit trail of all security updates
- **Consistent State:** All systems aligned with latest security posture

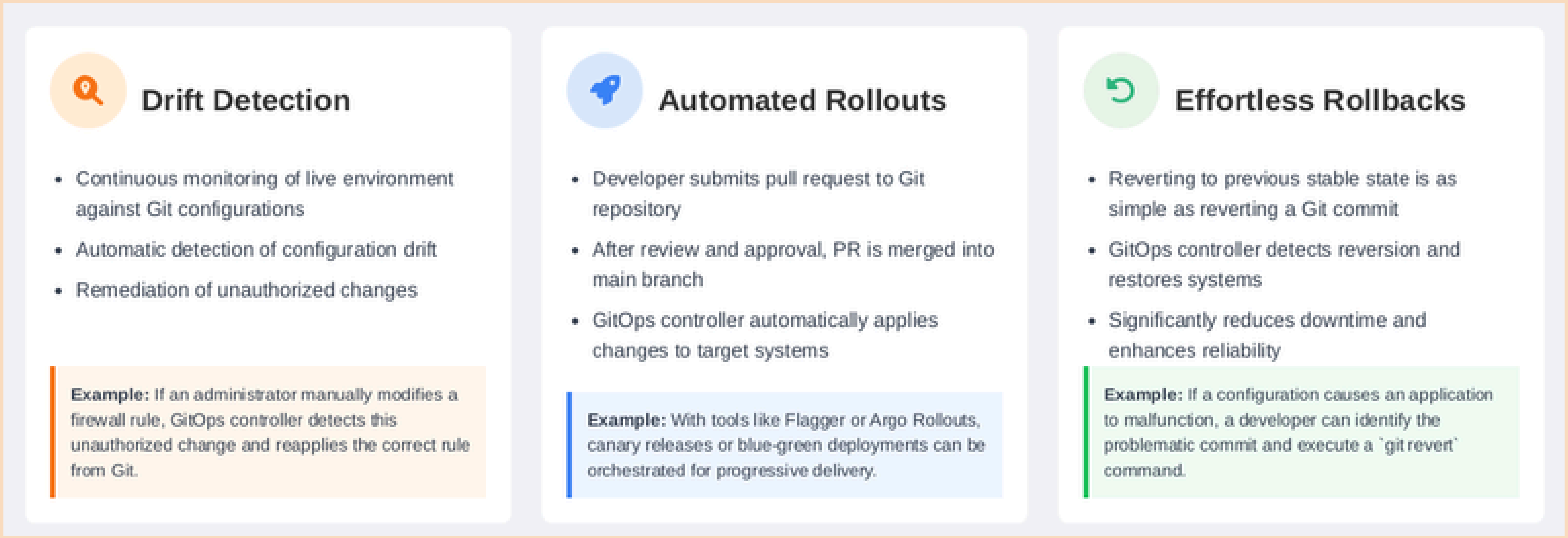
Gitops + Ubuntu Assembels



Building a Self-Healing Ubuntu Environment

- GitOps controller continuously monitors drift
- If config diverges, it auto-corrects to match Git
- System failures trigger rollback pipelines
- Enables resilience at OS level, not just app layer

Self Healing Infrastructure



Drift Detection

- Continuous monitoring of live environment against Git configurations
- Automatic detection of configuration drift
- Remediation of unauthorized changes

Example: If an administrator manually modifies a firewall rule, GitOps controller detects this unauthorized change and reapplies the correct rule from Git.

Automated Rollouts

- Developer submits pull request to Git repository
- After review and approval, PR is merged into main branch
- GitOps controller automatically applies changes to target systems

Example: With tools like Flagger or Argo Rollouts, canary releases or blue-green deployments can be orchestrated for progressive delivery.

Effortless Rollbacks

- Reverting to previous stable state is as simple as reverting a Git commit
- GitOps controller detects reversion and restores systems
- Significantly reduces downtime and enhances reliability

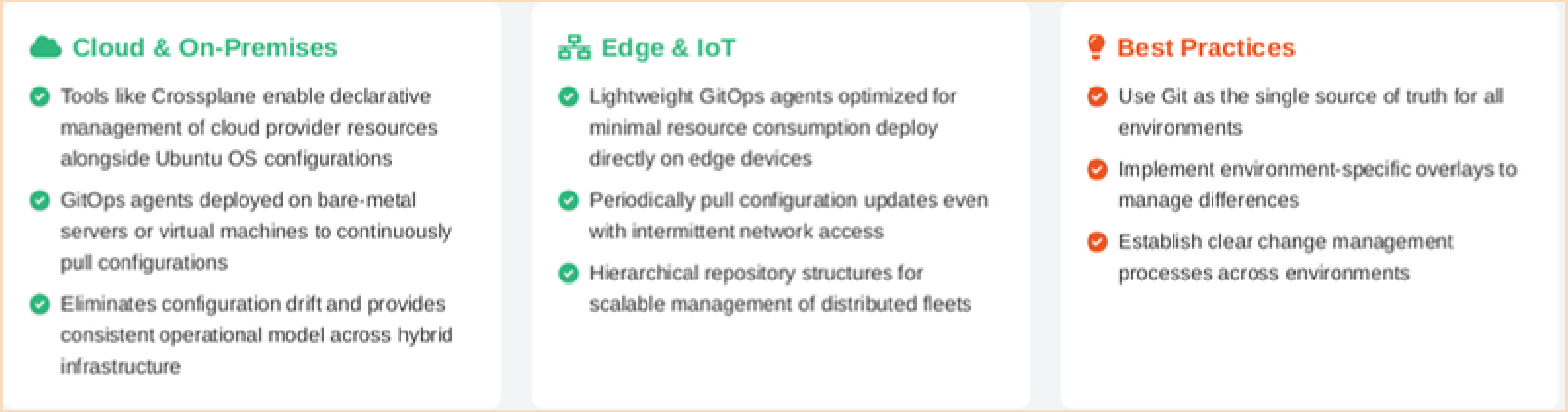
Example: If a configuration causes an application to malfunction, a developer can identify the problematic commit and execute a `git revert` command.

Demo or Example Architecture

/ubuntu/apt-packages.yaml
/ubuntu/snaps.yaml
/ubuntu/systemd/

- Git repository:
- Flux reconciles these to live Ubuntu instances via SSH or agent
- Git commit = automated rollout

Multi Environment Deployments



Cloud & On-Premises

- Tools like Crossplane enable declarative management of cloud provider resources alongside Ubuntu OS configurations
- GitOps agents deployed on bare-metal servers or virtual machines to continuously pull configurations
- Eliminates configuration drift and provides consistent operational model across hybrid infrastructure

Edge & IoT

- Lightweight GitOps agents optimized for minimal resource consumption deploy directly on edge devices
- Periodically pull configuration updates even with intermittent network access
- Hierarchical repository structures for scalable management of distributed fleets

Best Practices

- Use Git as the single source of truth for all environments
- Implement environment-specific overlays to manage differences
- Establish clear change management processes across environments

Bridging Cloud-Native and Ubuntu Ecosystems

- Common workflows for K8s, apps, and base OS
- GitOps unites platform, Dev, and Ops teams
- Enables one operational model across environments
- Ubuntu + GitOps = unified declarative operations stack

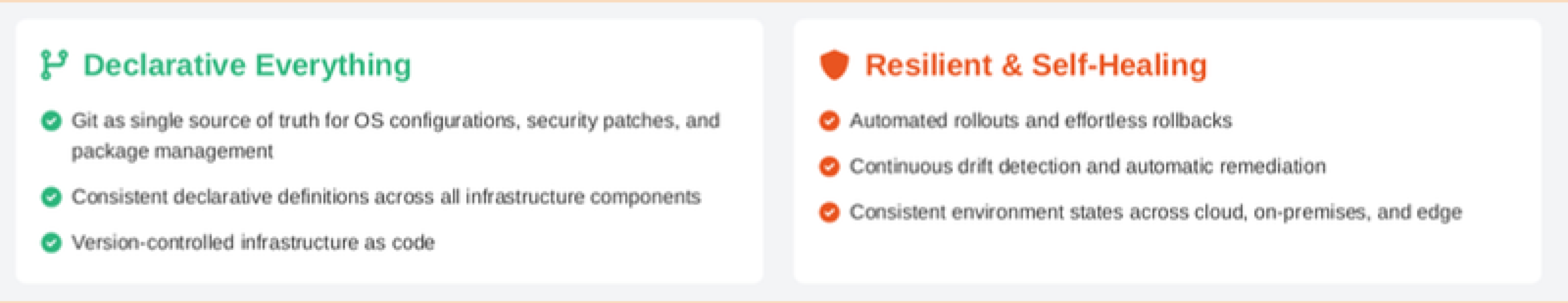
Key Takeaways

- ✅ GitOps applies to everything, not just Kubernetes
- ✅ Ubuntu configs, Snaps, and packages can be declaratively managed
- ✅ Automation ensures reliability, consistency, and rollback
- ✅ Git is your system of record across environments

Final Thoughts

- Declarative systems are the future of ops
- Ubuntu’s openness makes it GitOps-friendly
- Adopt Flux or ArgoCD today for your infra
- Let Git drive your infrastructure evolution

Future of Unified Infrastructure



Declarative Everything

- Git as single source of truth for OS configurations, security patches, and package management
- Consistent declarative definitions across all infrastructure components
- Version-controlled infrastructure as code

Resilient & Self-Healing

- Automated rollouts and effortless rollbacks
- Continuous drift detection and automatic remediation
- Consistent environment states across cloud, on-premises, and edge

Connect with me for any queries !

SCAN ME

UbuCon India 25

Thank You!

Qualcomm Canonical IT'S FOSS