**14. Enterprise Cloud Native Automation**

Shivakumar R Goniwada[**1**](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Aff2)

(1)

Bangalore, India

Software engineering’s main objective is to unify the development of software (Dev) and subsequent operations (Ops), and it is this combination of cultural values, practices, and tools that allow an organization to deliver software applications quickly.

DevOps is a set of rules, principles, or manifestos that are used to increase automation as the code is developed, built, and deployed. Many concepts that are part of the DevOps pipeline, such as continuous integration (CI) and continuous delivery (CD), are used by various teams that do not follow DevOps completely. A complete DevOps pipeline recommends process automation to be used from the discovery phase through the deployment, infrastructure, and operation phases.

DevSecOps is primarily the addition of security, performance, and stability to the DevOps cycle. DevSecOps is built on top of DevOps and adds extra checks and a shift-left approach at each stage.

In this chapter, I am not covering the entire DevSecOps story, because there are tons of books, articles and whitepapers available. The purpose of this chapter is to guide you through the best practices and how to leverage cloud development platforms on your journey of cloud native application development.

In this chapter, I will explain the following:

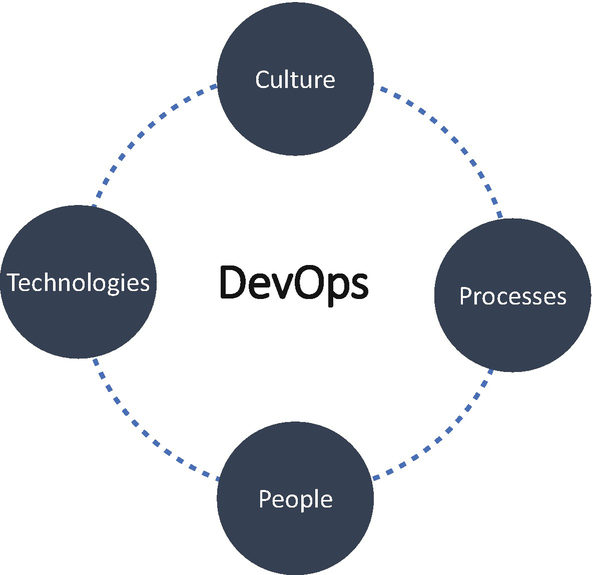
* DevSecOps pipeline
* DevSecOps and the cloud
* How to embrace a cloud development service to accelerate your development
* How to scale DevSecOps into your enterprise

**Introduction**

Innovation and continuity never stop. During the COVID-19 crisis, teams were sitting at home and working in silos in remote places. Businesses must ensure their workforce can still develop and deploy a solution even while remote, and the DevOps methodology addresses these challenges. It is a popular way of working in many businesses and provides a framework to coordinate your IT teams. It brings together your business, development, and operations teams, eliminating the barriers caused by physical location, organizational functionality, and business goals.

*Continuous* is one word that you will often use in your projects when discussing development, deployment, and operation. Almost everything in automation is continuous, whether it is continuous integration, continuous delivery, continuous testing, continuous infrastructure, continuous observation, and continuous operation.

As shown in Figure [14-1](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig1), DevOps is the main pillar of automation; it builds a culture of trust, collaboration, and continuous improvement. As a culture, it holistically views the development process and everyone involved, like developers, testers, operations, security, infrastructure, and client teams. DevOps is not just about tooling; it’s also about people working on a project.



***Figure 14-1***

DevOps pillars

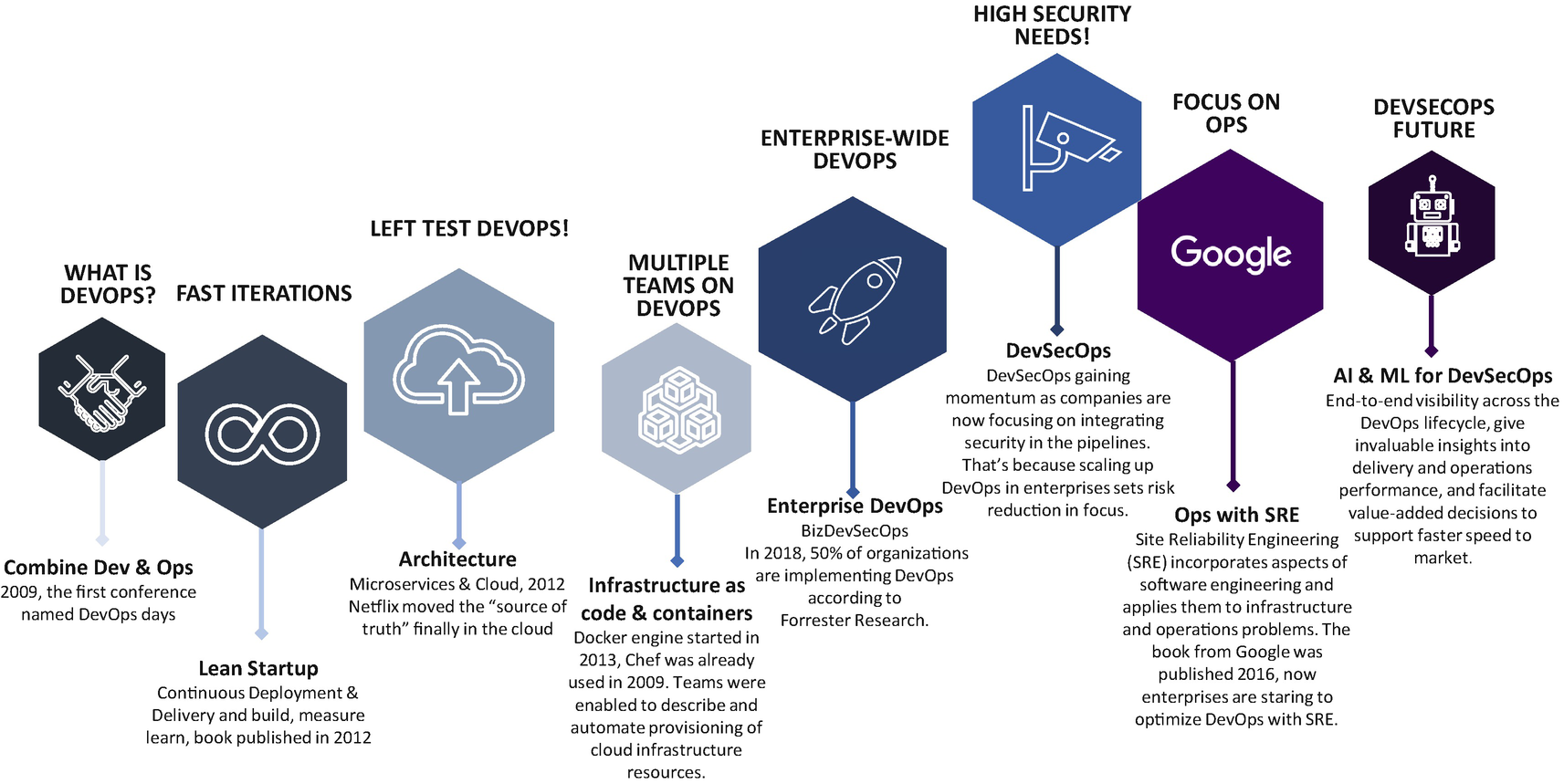
* *Culture*: There is strong communication and integration between all stakeholders.
* *Processes*: An automated deployment pipeline is integrated with security reviews and testing, with a strong feedback loop to the development team and operations.
* *Technologies*: There is an advanced combination of open source and commercial tools assessing various aspects of the application.
* *People*: It is a philosophy that focuses on engineers and how they can better work together to produce great software.

The DevOps culture brings nirvana in the development process. It helps the organization with the following:

* Faster time to market to gain market advantage
* High quality to detect failures to fix them early
* Adopting changes based on business demands
* Adopting changes based on technology evolution
* Effective collaboration and communication
* Integrating feedback effectively in the development process to get better
* Adopting improvement and innovations
* Avoiding an error-prone manual process
* Adopting shift left

**DevOps Today and Tomorrow**

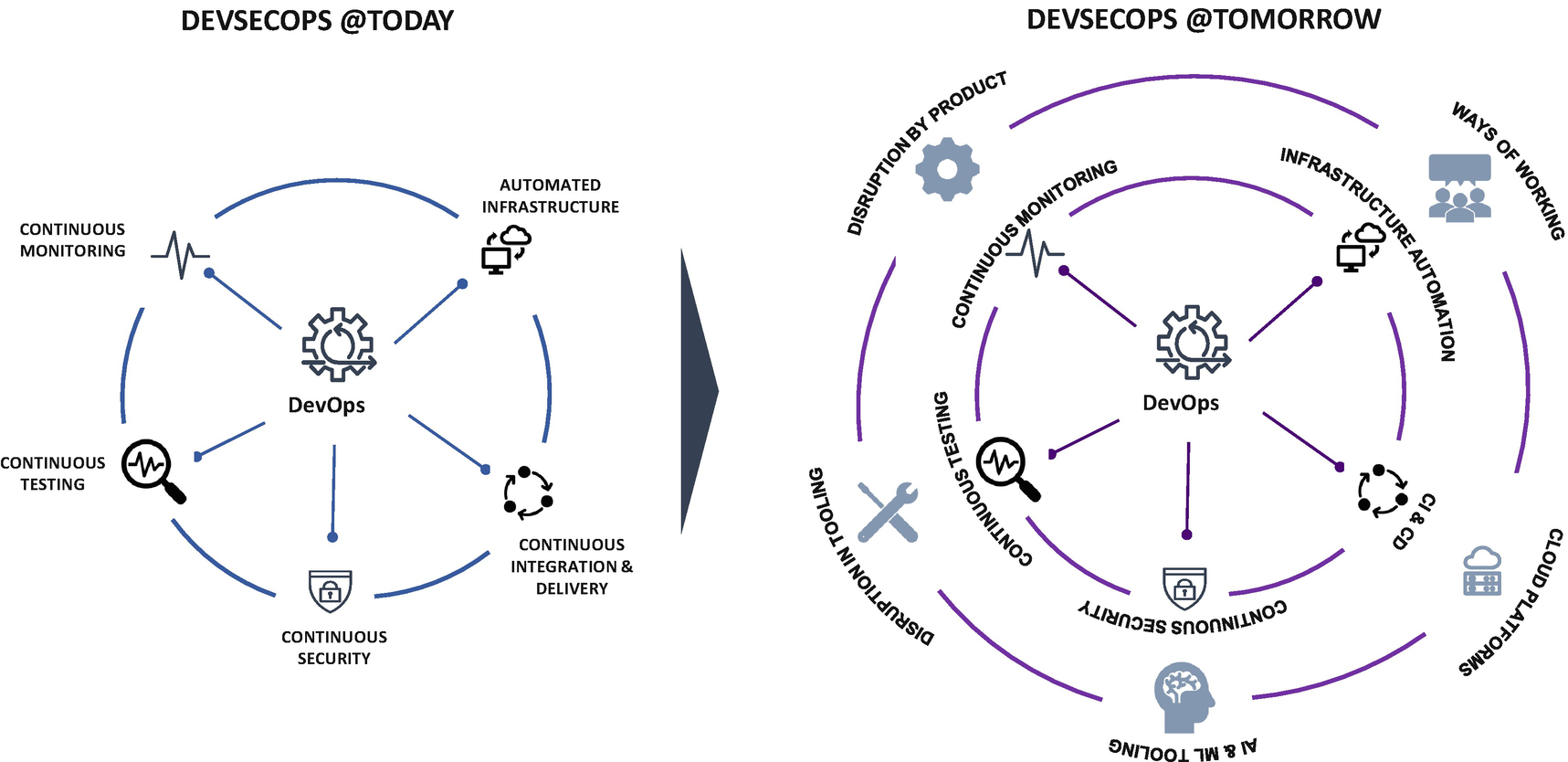
Coined in 2009, DevOps has evolved over the years. Figure [14-2](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig2) illustrates the journey of DevSecOps.



***Figure 14-2***

DevSecOps journey

As shown in Figure [14-3](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig3), today DevOps is about culture, automation, lean, measurement, and sharing. Most organizations have already adopted and matured using this model, due to technology disruptions and business disruptions. Today’s DevOps may not be able to meet disruptions in the future, though. Therefore, DevOps is rapidly branching into feature streams focused on cloud, security, data science, machine learning, artificial intelligence, and lean ways of working.



***Figure 14-3***

DevSecOps today and future

In present and modern-day architecture, with the adoption of multicloud and containerization, enterprises are required to embrace a new-age tool and learn new ways of working. It makes the DevOps pipeline crucial to a business, which needs to maintain a development pipeline with a new set of tools and quickly configure cloud development pipelines.

Here are some trends occurring in the world of cloud native and in regard to its relationship with DevOps:

* There’s an increase in the variety of cloud services that are leading to multicloud and hybrid cloud platforms.
* Data science and data integration are embedding data pipelines in the data lifecycle for speed and accuracy of the analytics and model management.
* More business involvement is required in software engineering along with the development team and operations teams. This requires a foster collaboration between all stakeholders so the organization can deliver software quickly and efficiently by using DevBizOps.
* This requires intelligence across the application lifecycle to focus on building and releasing best-in-class products by quickly using AIOps.
* Automation of the network is important along with infrastructure; therefore, you need to include DevOps best practices in network operations that drive network infrastructure as code to achieve faster delivery by using NetOps.

The future of DevOps requires “continuous everything.” This means that security, compliance, network, “-ilities,” and all other critical software components are automatically and continually implemented without compromising any release process.

**From DevOps to DevSecOps**

In DevOps, development teams are more agile. The product goes to market quicker, the team innovates faster, there is continuous everything, and there are measures everywhere, but security is still segmented and siloed away from the core software engineering functions. But as the number of cyberattacks increases, more compliance products are exposing all sorts of functionality across geographies, so security cannot remain separate from the DevOps process. Security must be integrated early and throughout the software engineering lifecycle.

You can enable DevSecOps by adding security compliance checks in the DevOps cycle. This helps you to address the principles of a single pane of glass, design for security, etc.

* Tightly integrate security tools and processes throughout the DevOps pipeline.
* Automate core security tasks by embedding security controls early on in the software development lifecycle.
* Implement continuous monitoring and remediation of security defects across the application lifecycle including development and maintenance.

The following are the benefits of DevSecOps in the pipeline:

* DevSecOps implements the secure by design principle by using automated security reviews such as static application security testing (SAST) and dynamic application security testing (DAST).
* Security issues are detected and remediated during development phases, which increase the speed of delivery and enhances quality of software components.
* In DevSecOps, security auditing, monitoring, and notification systems are automated and continuously monitored, which enhances the compliance in an application.
* By integrating security in software engineering and operation, engineering fosters collaboration across teams.

**Driver for Shift-Left Security**

The shift-left security approach has many benefits including cost efficiency, shorter release cycles, and better code quality, and it is able to provide the following:

* *Business risk reduction*: Test early on in the software engineering cycle, and address and prevent vulnerabilities before deploying to production; this significantly reduces business risk.
* *Faster release cycles*: Security testing, both SAST and DAST, should be integrated and automated as part of the pipeline to enable quicker release to the production.
* *Better code quality*: Addressing defects early or preventing defects from being introduced results in higher code quality.

**Automation Principles and Best Practices**

The principles and practices of DevOps help your enterprises to innovate with greater efficiency and agility. In this section, I will provide few practices and identify problems they eliminate. Along with this, I will touch upon cloud principles that help to achieve continuous operation in public, private, and hybrid clouds.

* *Collaborative environment*: Use the right tools to enhance the collaborative environment and create the right communication among all stakeholders.
* *Eliminate waste*: Eliminating waste is important in lean processes. Unnecessary functionality, code, or effort is wasteful. Delaying the delivery of value to customers and inefficient processes are other examples of software development waste.
* *Adopt agility and focus more on automation*: Adopt agile development methodologies as explained in the previous chapter and follow the automation in every cycle of software engineering.
* *Focus on shortened feedback loops*: Start the feedback mechanism early in the software engineering lifecycle.
* *Create knowledge*: Teams share knowledge within the team and across teams, through code reviews, documentation, learning sessions, training, and collaboration with Confluence tools that can be used as a knowledge database.
* *Common and shared goals across all stakeholders*: Ensure that the entire team and all relevant stakeholders, including business owners, are engaged in deriving common and shared goals.
* *Shift-left security*: Enable an end-to-end DevSecOps capability by integrating security earlier in the software development and delivery.
* *Everything as code*: Enable consistency, automation, and repeatability by adopting an as-code approach across the DevSecOps pipeline encompassing infrastructure, configuration, security policies, compliance validation, and testing as code.
* *Tooling optimization*: Optimize the use of all DevSecOps support tooling. Enable logging whenever possible to get a holistic view of the pipeline and application. Ensure testing remains in compliance with regulatory requirements.
* *Self-service*: Enable self-serviceability with users provisioning their services (i.e., compute, storage, environments) and empower them with tools to make low-impact changes directly without IT support.
* *Governance approach*: Adopt federated governance based on outcomes. Track and measure DevOps. Utilize enterprise-wide KPIs to monitor progress, applied to both DevOps and traditional SDLC so the performance improvements of DevOps will be measured.
* *Continuous improvement*: There is no full stop on anything; it is a continuous evolution. Therefore, focus on hypothesis-driven improvements and optimization of flows.
* *Deployment process*: Adopt a zero-touch, zero-downtime deployment with A/B testing enabled and automatic rollback of failed changes.

**Site Reliability Engineering**

Site reliability engineering (SRE) creates a bridge between development and operations by applying software engineering best practices. SRE was first introduced in 2003 by Google engineers. SRE is way of thinking about and approaching production. It is a set of principles and practices. SRE is aimed at developing automated solutions for operational aspects such as monitoring, performance, capacity planning, and disaster response.

SRE helps teams to determine what new features are launched and when by using SLAs to define the required reliability of the system through SLI and SLO.

DevOps is an approach to culture, automation, and platform design that delivers increased business value and responsiveness through high-quality service delivery. SRE can be considered an implementation of DevOps. Like DevOps, SRE is about team culture and relationships. Both SRE and DevOps work to bridge the gap between the development and operations teams to deliver higher quality and faster services.

According to Google, the following are the types of SRE team implementation:

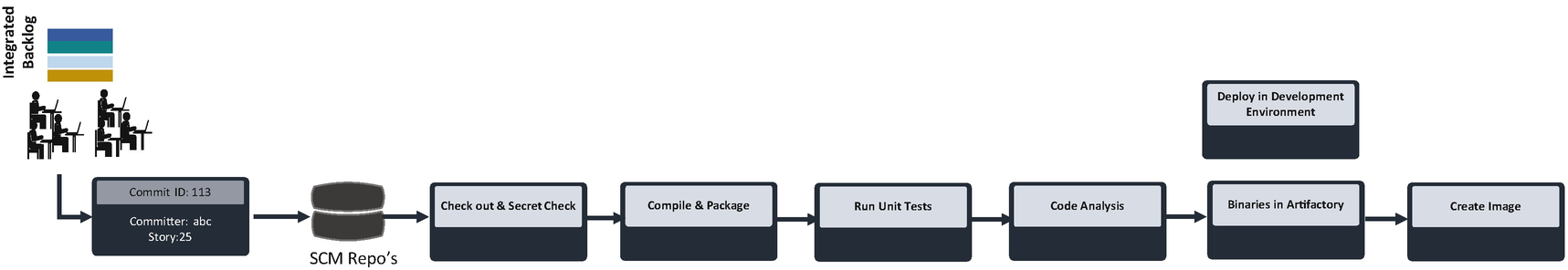
* *Kitchen sink, aka “everything SRE”*: This describes an SRE team where the scope of services or workflows covered is usually unbounded.
* *Infrastructure*: These teams focus on behind-the-scenes efforts. A common implementation includes maintaining shared services (such as a Kubernetes cluster) or maintaining common components (like CI/CD, monitoring, etc.) built on top of the cloud.
* *Tools*: A tools-only SRE team tends to focus on building software that help engineers to measure, maintain, and improve system reliability.
* *Product/application*: The SRE team works to improve the reliability of a critical application or business area.
* *Embedded*: SRE is embedded with the development team, usually one per developer team.
* *Consulting*: SRE is similar to consulting work of any organization, but the difference is that the consulting SRE team tends to avoid changing the customer code and configuring the services in scope.

**DevSecOps**

This section covers DevSecOps in more detail.

**Continuous Integration**

As shown in Figure [14-4](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig4), CI is the base of the DevSecOps culture of transformation that automates the integration of code changes from multiple pod teams into a single software project. CI is the basic pipeline of the entire DevSecOps adoption. The primary benefit of adopting CI is that it saves you time during development by automating your code merges, unit tests, code reviews, and builds.



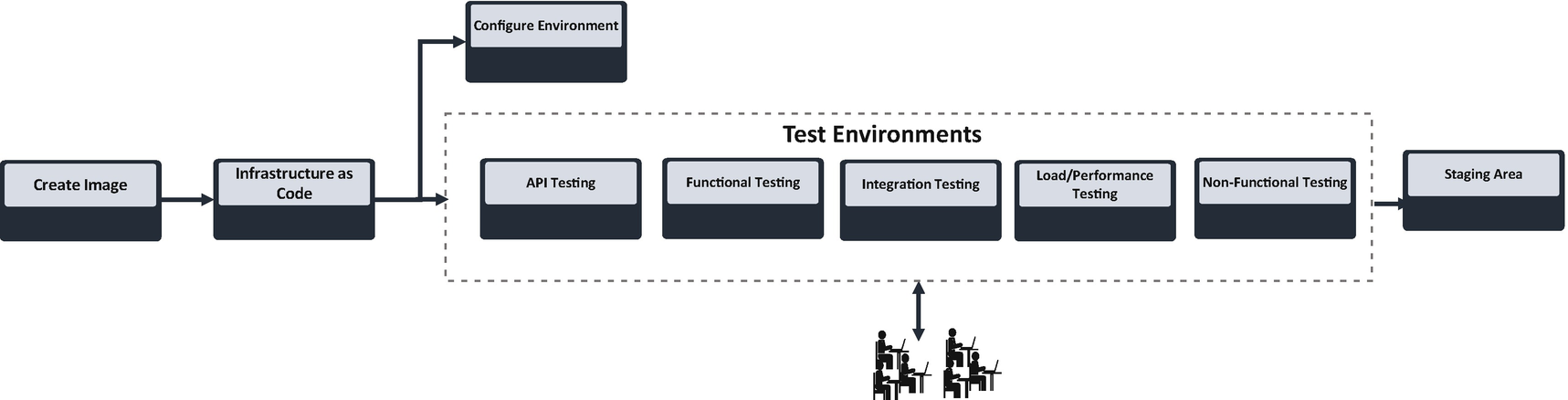
***Figure 14-4***

Continuous integration

CI involves making incremental code changes and continuously integrating on a frequent, regular basis. In this process, small changes are made to code by an engineer, and that code is subsequently checked into the source code repository. When the code is checked in, an automated build is typically triggered.

**Continuous Delivery**

As shown in Figure [14-5](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig5), CD is the ability of an organization to release changes to users quickly and in a sustainable and repeatable way. When CI completes, the CD begins. It essentially automates the delivery of applications to specific environments including the development, testing, and production environments.



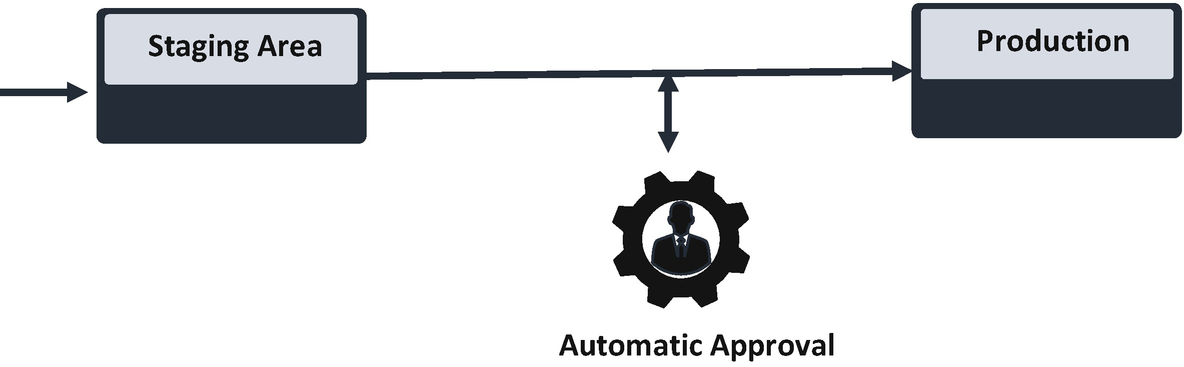
***Figure 14-5***

Continuous delivery

CD helps you to automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying them into an environment. These tests include API, UI, load, functional, integration, reliability testing, etc. This helps your team to thoroughly validate updates and pre-emptively discover issues.

**Continuous Deployment**

As shown in Figure [14-6](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig6), continuous deployment (CD) takes continuous delivery (CD) one step further. In continuous deployment, all the code is built and tested and then pushed to nonproduction environments. There can be multiple parallel and various testing before certifying the quality of software.



***Figure 14-6***

Continuous deployment

In continuous deployment , the software is delivered to the staging area along with test automation. When done properly, the software application should be in a state that it can be deployed to production at any time. Continuous deployment merely automates the final step so that all changes are automatically deployed to the production environment. Practically deploying to production depends on the software type and organizational maturity because it requires certain approvals from relevant stakeholders. In continuous deployment, you are going to automate the approval and push binaries into the production environment.

**DataOps**

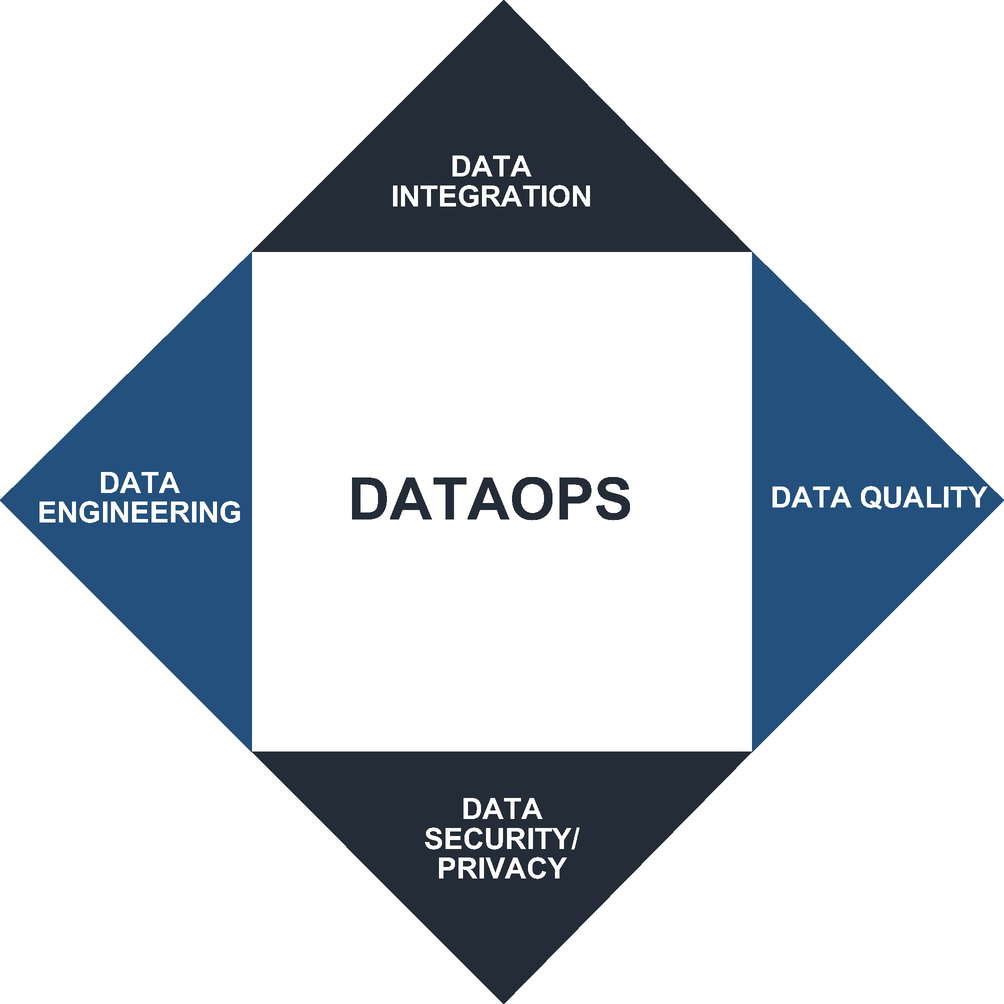
*“DataOps is a collaborative data management practices focused on improving communication, integration, and automation of data flows between data managers and data consumers across an organization.”—Gartner*

*“The goal of DataOps is to create predictable delivery and change management of data, data models, and related artifacts. DataOps seeks to reduce the end-to-end cycle time of data analytics, from the origin of ideas to the literal creation of charts, graphs, and models that create values.”—Gartner*

DevOps analytics turns data from DevOps tools into insights that aid in decision-making. It also gives stakeholders visibility into various DevOps practices, helping you to identify strengths and opportunities for improvement across every aspect of the adoption process. For example, the adoption owners can find the root cause of a bottleneck in software agility much faster among large application portfolios using DevOps analytics.

It is an automated, collaborative, and agile practice to improve the quality and eliminate waste, bottlenecks, and inefficiencies in the data lifecycle. It breaks data silos and rapidly meets new business demands.

Figure [14-7](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig7) illustrates a DataOps strategy that is part of the DevOps pipeline and strives to speed up the production of data integration, data engineering, data quality, and data security/privacy. It accelerates the data lifecycle to reduce the time for data analytics.



***Figure 14-7***

DataOps pillars

The goal of DataOps is to streamline the design, development, and maintenance of applications based on data and analytics. It seeks to improve the way the data is managed and products are developed and coordinates with all the relevant stakeholders.

One thing you need to remember is that DataOps is not just DevOps applied to data analytics. DataOps communicates the data analytics to achieve what software engineering wants to attain with DevOps. DataOps can yield an order-of-magnitude improvement in quality and cycle time when data teams utilize new tools and methodologies.

**DataOps Principles**

The following are a few DataOps principles defined by the DataOps Manifesto. These principles help you to configure DataOps as part of the DevSecOps pipeline.

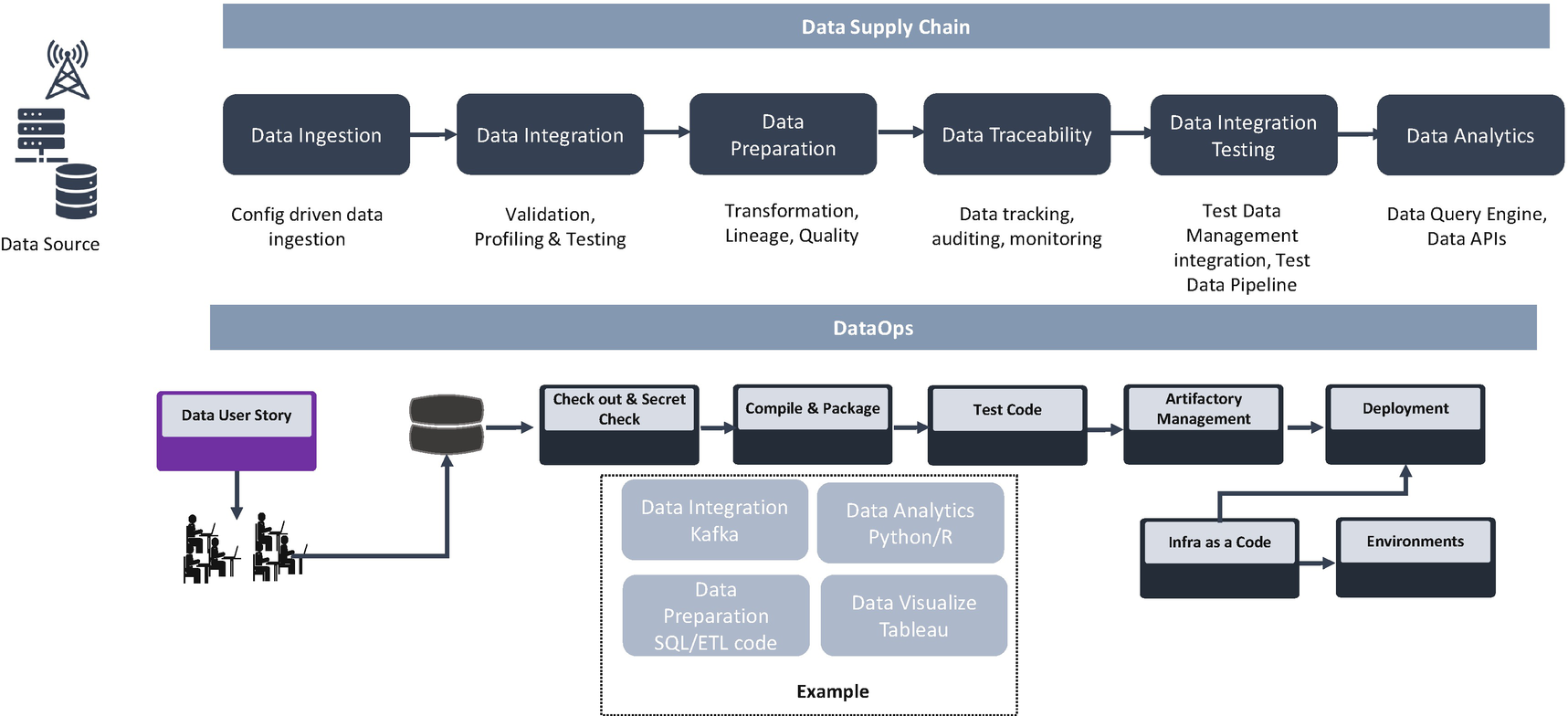
* *Value working analytics*: This primary measure of data analytics performance is the degree to which insightful analytics are delivered, incorporating accurate data atop a robust framework and system.
* *Continuous interactions*: Customers, analytics teams, and operations must work together continually throughout the project.
* *Self-organize*: Analytics insight, algorithms, architectures, requirements, and designs are well-defined by a self-organized team.
* *Analytics is code*: DataOps uses a variety of available tools to access, integrate, model, and visualize data. At a basic level, these tools generate code and configurations that describe the action taken upon data to deliver insight.
* *Version everything*: You need to reproduce the result, so version everything.
* *Quality*: The pipeline should be built with a foundation capable of automatically detecting irregularities and security issues in code.
* *Improve cycle time*: You should strive to minimize the time and effort to turn a customer’s need into an analytic idea. Create it in development, release it as a repeatable production process, and finally refactor and reuse the product.

**DataOps Pipeline**

DataOps is an operation for data analytics and works similarly to DevOps. It can yield an order-of-magnitude improvement in quality and cycle time when data teams utilize new tools and methodologies. As you already know, DevOps optimizes the application software engineering delivery and deployment. Similarly, DataOps optimizes analytics software in data engineering delivery and data operation. DataOps includes DevOps and other methodologies that apply to managing the enterprise data operations pipeline.

DataOps builds upon the DevOps development model, as shown in Figure [14-8](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig8). DevOps works on continuous integration with the build, check, and continuous delivery with automated tests. Similar to DevOps, the DataOps orchestrates the data pipeline from the data ingestion to data analytics, and the pipeline consists of many steps like Data Ingestion, Data Integration, Data Preparation etc as shown in Figure [14-8](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig8). An orchestrator is a tool that controls the execution of each step as shown in Figure [14-8](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig8).

For example, the orchestrator may create containers and invoke runtime processes like machine learning models to analyze data, transfer data from one step to another, and monitor pipeline execution.



***Figure 14-8***

DataOps pipeline

The data supply chain represents the flow of data from source to consumer by using various stages.

* *Data ingestion*: This includes the inputs into the data supply chain from a source like social medial, IoT, CRM, etc.
* *Data integration*: This includes integrating data from the identified data source. This integration could happen by using various protocols.
* *Data preparation*: Clean, enrich, standardize, and transform data and prepare data to make it business consumption ready.
* *Data traceability*: Trace data for auditing and monitoring. This traceability could happen by using various data monitoring tools.
* *Data integration testing*: Use the DataOps pipeline to test the data.
* *Data analytics*: Explore data, conduct analysis, and discover patterns. This could happen by using ML tools and exposing analyzed data to the consumer for decision-making or could feed into services and applications by using APIs and messaging software.

The previous example provides a clear picture of what probable software/code is used as part of the DataOps pipeline.

The DataOps pipeline uses the DevOps process to build, test, and deploy in the environment.

* *Compile and package*: The pipeline compiles the ETL app code and ETL pipeline as code and uses the streaming and batch processing of data.
* *Test code*: The pipeline tests data for quality measures, data profiling, data cleansing, data validation, and data reconciliation. The test monitors data values flowing through the data supply chain to catch anomalies or flag data values outside statistical norms. In DataOps, you need to conduct a test at every stage of the data supply chain.
* *Infrastructure as code*: In DataOps, build and test fall under CI, and deployment is CD. So, infrastructure as code is coming under CD. By using this method, you can create templates and configurations to provision infrastructure and deploy your data code.

**DevNetOps**

This section covers DevNetOps.

**Network Operation and Challenges**

The traditional network is hardware-based, proprietary, expensive, and difficult to scale. It has complex lifecycle management practices with rigid configurations. Traditional networks do not work for modern-day architecture and business.

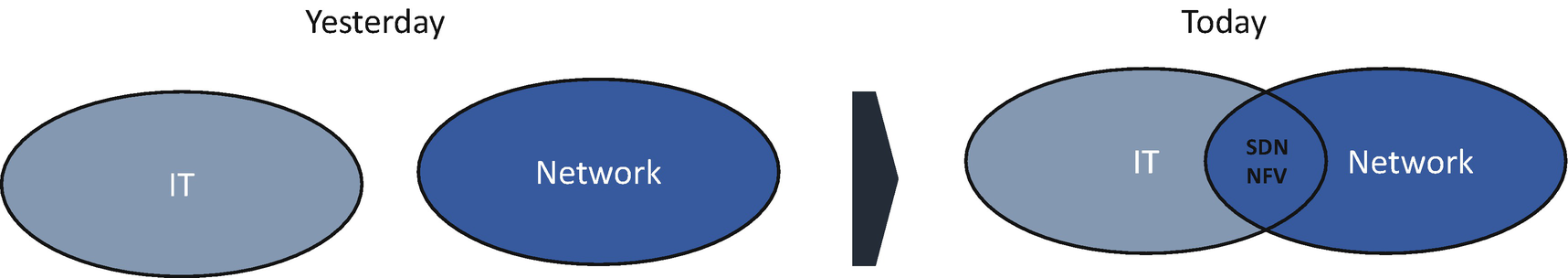
Today’s business requires a faster time to market with more services at a lesser cost and requires demand-based scalability. The team needs to focus on delivery with an innovation culture that helps to stay competitive among other teams.

To overcome these hindrances and to support modern business, we require a software-defined and highly configurable network. The software-defined network (SDN) is a new architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today’s application.

The SDN architecture consists of the following:

* *Network programmable*: It has a centralized control plane to control or program network devices using software applications.
* *Logical separation*: The network control plane and data plane are separate.
* *Centrally managed*: Network intelligence is logically located centrally in SDN controllers.
* *Network abstraction*: The application will interact with the network through APIs, instead of management interfaces tightly coupled to the hardware.
* *Open architecture and vendor neutral*: Network services and applications can run within a common software environment with interoperability support for multivendor network devices.

As illustrated in Figure [14-9](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig9), in the past the network and IT team worked in silos by using a different set of tools and unmatched deployment schedules. Earlier there was no universal open architecture, which made teams learn vendor-specific hardware, technologies, etc. All of this leads to larger capex and inefficient opex spend.



***Figure 14-9***

Journey toward a modern network

Today, IT and network teams have a common way of working and collaborating by using the open architecture across network vendors. This makes life simple and removes vendor-specific training and teams.

Virtualization and SDN help to reduce the variation in capex, optimize hardware usage, and share resources to reduce hardware maintenance costs.

**Why You Need DevNetOps?**

The virtual network functions (VNFs) are virtualized network services running on open computing platforms formerly carried out by proprietary, dedicated hardware technology. Common VNFs include virtualized routers, firewalls, WAN optimization, and network address translation (NAT) services. Most VNFs are run in VMs on common virtualization infrastructure software such as VMware or KVM.

* There are no standard procedures to develop and benchmark VNFS.
* There are no standard architectural guidelines for VNFs.
* Manual configuration, updating, and testing of VNFs is time-consuming.
* There are no standard protocols or configuration policies for VNF across vendors.
* Service providers have their workflow in infrastructure.
* There are no common KPIs defined for realizing NFV implementation success.

DevNetOps enables agility and quality in the following ways:

* Implements network as code and agile change management config + templates + artifacts + OS
* Implements a pipeline of continuous integration and testing, staging simulation, and delivery
* Orchestrates deployments, rolling upgrades, and traffic management
* Implements resilience and testing drills (Chaos Monkey)

**Network Reliability Engineering**

Network reliability engineering (NRE) is an emerging approach to network automation that stabilizes and improves reliability while achieving the benefits of speed. NRE is like SRE.

DevNetOps helps NRE to easily deploy, configure, validate, and certify with simple steps of execution. Just like SREs define their methods using DevOps, DevNetOps is a method that embraces NRE.

The following are the NRE principles that are derived from DevOps:

* Enable automation
* Orchestration transparency
* Continuously evolve
* Monitoring metrics

NRE keeps the reliability of the network as the topmost priority along with these qualities: agility, security, velocity, efficiency, and performance.

The NRE includes the following:

* *Code*: Using infrastructure as code and developing network code with versioning
* *Automation*: Using DevNetOps for automating and dynamic provision
* *Test*: Continuous automatic testing to meet SLAs
* *Monitor*: Monitoring the entire network infrastructure

The following are the activities of NRE:

* *Code*: The NRE team develops the network software artifacts, secrets, and configuration-as-a-service code and checks it in to the version control similar to developer check-in to a version control like GitHub or Bitbucket.
* *Build and deploy*: NRE automates versioned deployments, peer reviews, and testing. It automates provisioning of networking resources and configuration of the networks.
* *Test*: Through automation, staging, stress testing, and chaos engineering, an NRE ensures that the deliveries are reliable enough to meet service level objectives.
* *Monitor*: An NRE monitors service level indicators (SLIs), both manually and automatically with analytics that trigger automatic responses and alerts.
* *Measure*: Use indicators to measure their effectiveness in meeting reliability goals, such as mean time between failure (MTBF) and mean time to repair (MTTR).

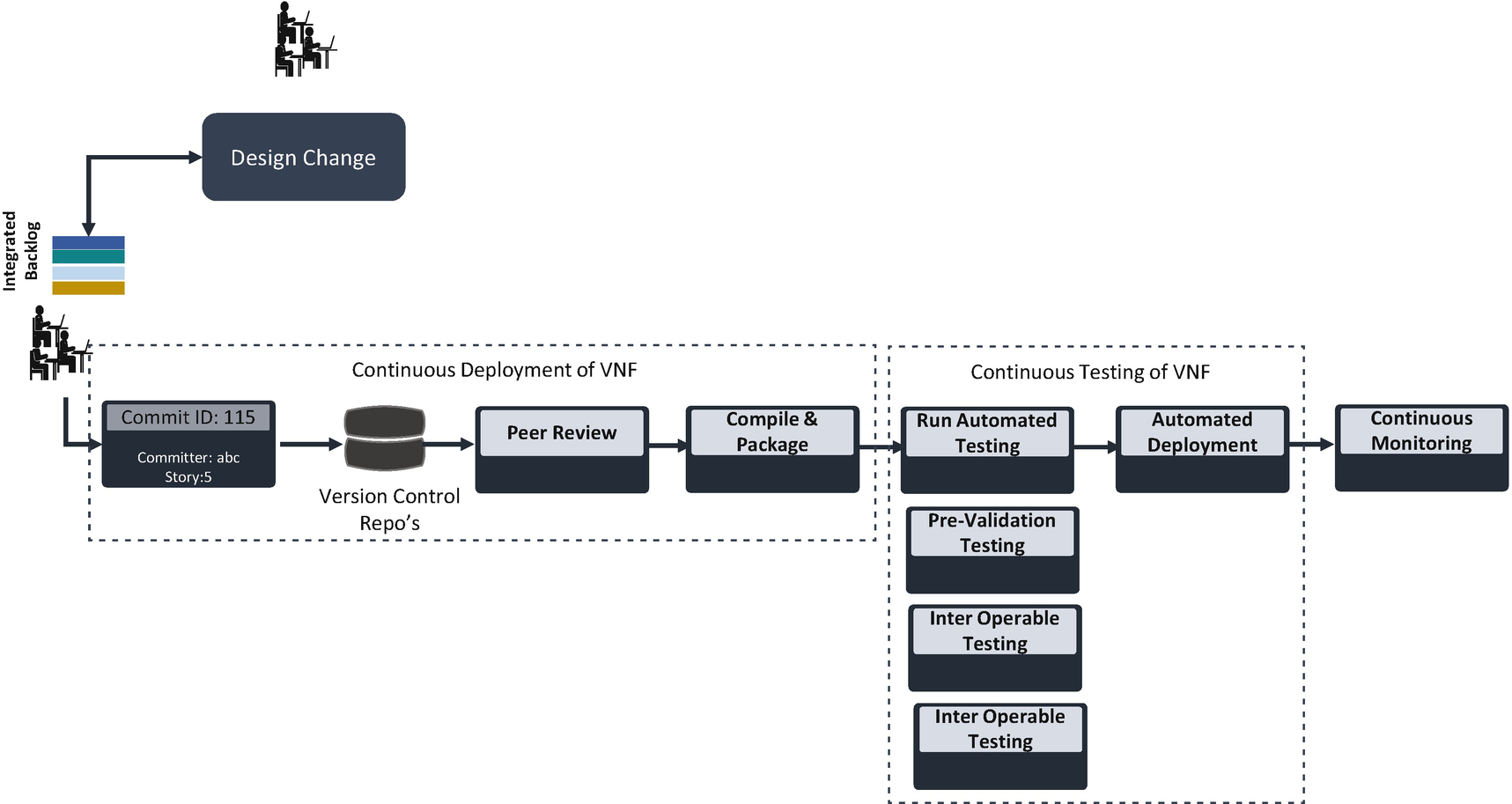
In the age of modern technology with technology and business disruption, your network must be able to support the application with speed and reliability. You cannot achieve these in the big-bang, manual approach; you need to have incremental development and automation to support this change. DevNetOps philosophies, culture, and automation will support these requirements.

**DevNetOps Pipeline**

As shown in Figure [14-10](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig10) and detailed in Table [14-1](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Tab1), DevNetOps is like the DevOps philosophies, culture, and behaviors of network operations (NetOps). DevNetOps helps NRE to easily deploy, configure, validate, and certify the steps of execution.

The DevNetOps provides the following:

* *Scalability*: It allows for optimized capacity on demand.
* *Agility*: It allows for simple configuration changes/updates and frequent upgrades with a shorter cycle of deployments and test.
* *Speed*: It allows for a faster time to market and updates with fewer technology disruptions.
* *Reliability*: It allows for deploying quickly and continuously with no failures and portability to any environment.
* *Security*: Network and security should co-exist for the safety of products, vendors, customers, and also end users.
* *Simple*: It allows for configuration as code and service handling complexity with less human error.



***Figure 14-10***

DevNetOps pipeline

***Table 14-1***

Network Pipeline Details

| Pipeline Steps | What Tools Can Be Considered? | What Process Followed? | Who Are Involved? |
| --- | --- | --- | --- |
| Network as code | Git, GitLab, Bitbucket, Gerrit  Infrastructure as code (IaC) tooling for the cloud  Declarative config as code (YAML)  Actual code is programmed by extension | Agile methodology  Reviewing  Design templates | Developers  Network team |
| Pipeline orchestration | Build (for example, Aminator or Packer)  Testing (for example, Open Stack, MANO)  Orchestrate deployment (for example, Spinnaker) | Continuous integration and delivery  Automatic and manual check  Continuous deployment | Test-driven development  Ops specialist |
| Micro-immutable architecture | Containers and functions  Container as a service (CaaS) and function as a service (FaaS) to run SDN system  Secrets, configs, volumes | Design/package software into a single-purpose service | Network architecture team |
| Resiliency design and drills | Net Chaos Monkey and watchdogs  Kill -9 command, unplug, or cut cables | Develop stress for staging  Run periodically in production | Performance engineering team  Resilience team |
| Measurement | Dashboards  KPIs | Incidents playbooks | Operation team |

**DevOps in the Cloud**

Hosting DevOps in the cloud can help an organization evolve from a reactive to a proactive approach. Whatever circumstances you are operating in, a cloud DevOps solution provides a business with a way to accelerate its software development and delivery.

A cloud DevOps solution is cloud native, and by adopting it, your organization can achieve delivery through continuous integration, continuous delivery, and continuous deployment with the required level of services and testing to deploy quality solutions to customers.

DevOps on the cloud provides a few benefits over a traditional on-prem solution:

* *Backup as a code*: Backup restoration is automated in the cloud, allowing your engineers to integrate backup as code with a continuous integration stack to automate, restore, and delete backups.
* *Business agility*: Cloud DevOps solutions can be seamlessly integrated into multiple business units. It is easy to set up and configure CI and CD pipeline tools.
* *Continuous monitoring*: Cloud services provide monitoring and observability services, and these services are very well integrated with the DevOps pipeline to monitor services. These services provide actionable insight to monitor applications, optimize resource allocation, respond to performance changes, and offer an integrated dashboard to keep track of application, infrastructure, and security.
* *Infrastructure automation*: The open source PaaS services manages multicloud infrastructure and automation management.
* *Configuration as a code*: DevOps supports configuration across the DevOps lifecycle including continuous integration, continuous delivery, continuous deployment, and infrastructure as a code. It is a string of YAML code or scripts that standardize the configuration of the network, server location, etc.

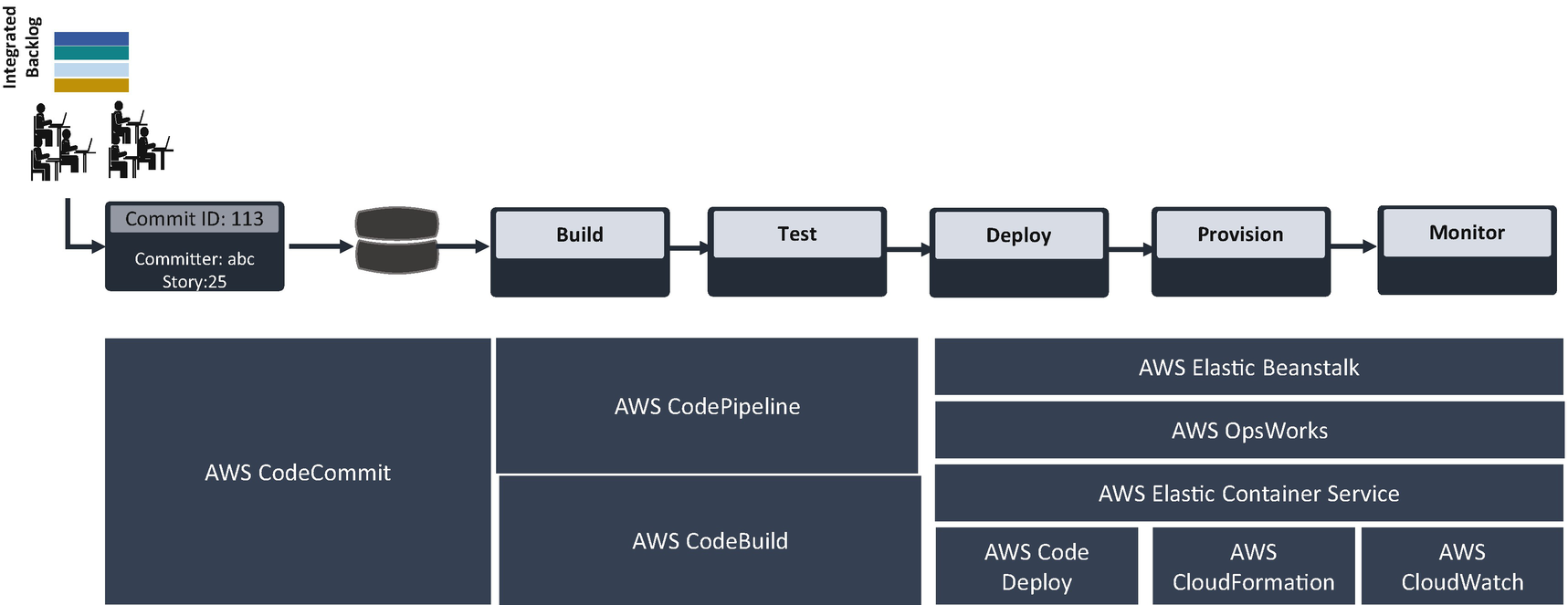
**AWS Cloud**

The AWS DevOps services provide a more reliable and quicker configuration of tools. These services simplify provisioning and managing infrastructure, deploying application code, automating the software release process, and monitoring your application and infrastructure performance. The AWS services are preconfigured, and there is no setup or software to install. This reduces your time to configure the DevOps pipeline and day-to-day operation of the pipeline. These services are built in to the cloud and manage single instances or scale to thousands depending on your volume of services. These services are linearly scaled based on your application.

AWS helps you to use automation so you can build faster and more efficiently. Using AWS services, as shown in Figure [14-11](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig11) and Table [14-2](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Tab2), you can automate manual tasks or processes such as deployments, development and test workflows, container management, and configuration management. You can set up an access control mechanism for your DevOps services by using identity and access management with your AWS accounts.

AWS has a larger partner ecosystem integrated into AWS services. You’re open to use any third-party services with AWS services. For example, if you want to use multicloud infrastructure automation, you can use Terraform services to automate, and if your team wants the source code to be on-prem with Bitbucket or GitHub, you can use AWS services for the rest of the pipeline.

AWS provides a pay-as-you-go model, so you need to pay only for the duration of usage.



***Figure 14-11***

AWS development services

***Table 14-2***

Comparison Chart with AWS DevOps Tools

| AWS DevOps Tools | Open Stack Tools |
| --- | --- |
| High availability and durability included. | Tools supports high availability and durability, but you need design infrastructure to support these features. |
| Easy access and integration with other services. | Tools will include AWS service plugins. |
| No provisioning servers or patching software. | Servers have to be configured and require regular patching. |
| Build servers are auto-scaled, and pricing is pay for what you use. | Scaling can be achieved by adding additional containers. |
| AWS tools are limited to AWS and do not have support now AWS tools and services. | Open stacks are matured, and more plugins are available and work for a multicloud environment. |
| Integrates with AWS Identity and Access Management (IAM) for roles and access management. | Integrates with any other IAM for roles and access management. |
| AWS services are pay-as-you-go model. | Requires additional enterprise support services. |

**Azure Cloud**

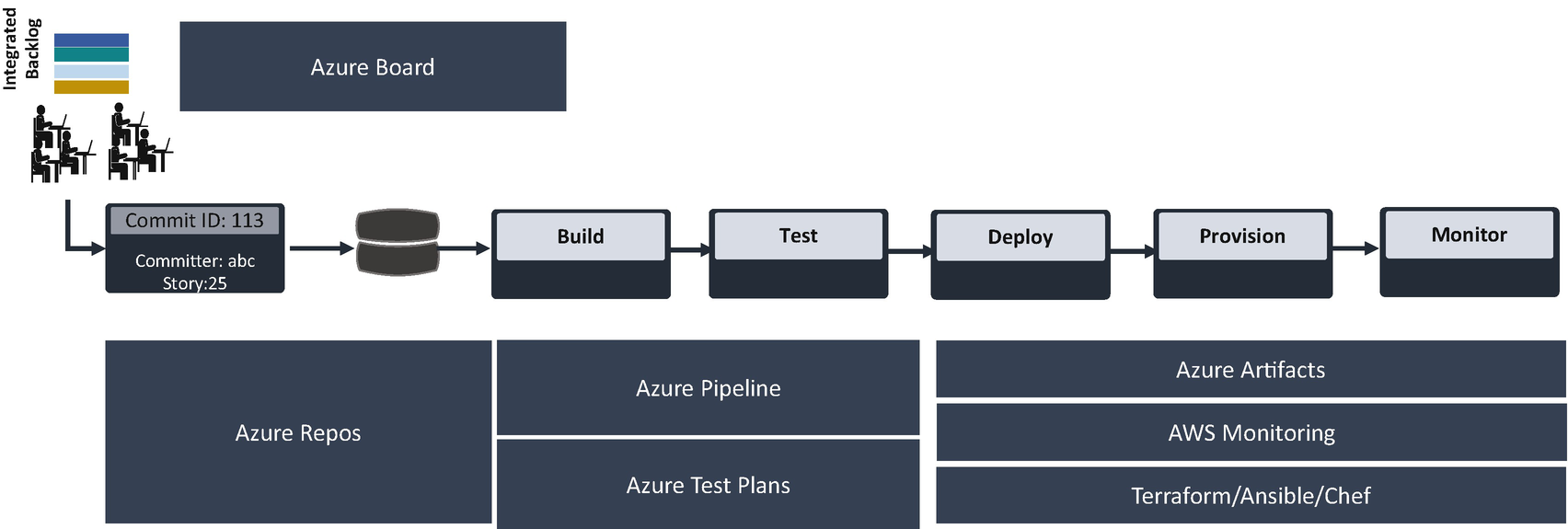
Azure DevOps is a Microsoft product that provides development services to support teams to plan work, collaborate on code development, and build and deploy applications.

Azure DevOps is offered in two forms.

* *Azure DevOps Server*: Previously known as Team Foundation Server (TFS), this is an on-premises offering.
* *Azure DevOps Service*: This was previously known as Visual Studio Team Services; it provides a PaaS-based offering to manage the end-to-end DevOps lifecycle.

The Azure DevOps service provides a platform for implementing the DevOps process across different IT segments. This tool supports various practices under DevOps such as continuous planning, continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring. These tools support integration with various other open and commercial tools such as code analysis tools, security tools to scan vulnerabilities in code, and infrastructure provisioning tools to automate the infrastructure such as Terraform and Ansible Tower.

As shown in Figure [14-12](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig12), the Azure DevOps multistage pipeline provides an easy way to use templates to configure CI and CD pipeline. The multistage pipeline provides features to add extensions such as build quality checks, security checks, infrastructure provision, etc.



***Figure 14-12***

Azure development services

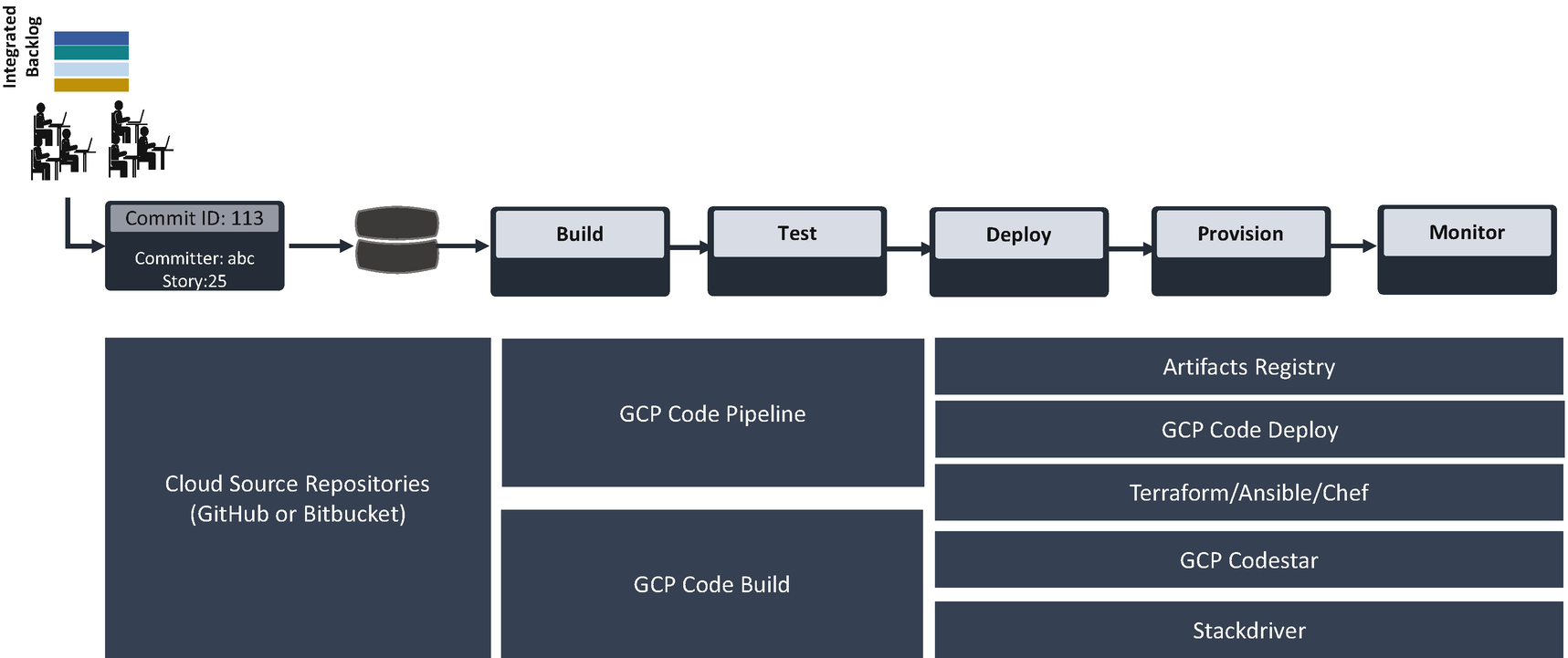
The following are the Azure DevOps tools that are provided to set up a DevOps pipeline in the Azure cloud:

* *Backlog and user story*: Azure Board can help your teams to manage software projects. It provides a rich set of capabilities including native software support for Scrum and Kanban, customized dashboards, and integrated reporting. You can easily start tracking user stories, backlog items, tasks, and bugs associated with your project.
* *Source code management*: Azure Repos is a set of version control tools that you can configure for your source code. It supports Git and Team Foundation Version Control (TFVC).
* *Build and release*: Azure Pipeline is a cloud service that you can use to automatically build and test your code. It combines CI and CD to test and build your code constantly and consistently.
* *Test*: Azure Test Plans allows you to create test plans and test cases. It supports both automated and manual testing.
* *Binary package*: Azure Artifacts stores the compiled code and other dependent binaries with version control.

**Google Cloud**

GCP provides a vast number of services for a cloud native application. Apart from these services, GCP supports a lot of DevOps and SRE tools that make the process speedier and deliver the services more reliably.

GCP supports DevOps efforts by providing services to build, store, and deploy apps. Figure [14-13](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig13) are the services that you can use in your software engineering lifecycle.



***Figure 14-13***

GCP DevOps pipeline tools

* *Artifact registry*: It enables you to centrally store binaries and build dependencies. It is a central location for storing packages and Docker images.
* *Software release workflow*: The GCP Code Pipeline service is a CI and CD tool for fast, reliable application and infrastructure updates. The Code Pipeline builds, tests, and deploys code every time on the system when there are changes and based on the release process models defined.
* *Build and test code*: The GCP Code Build service executes your builds on GCP. It imports source code from cloud storage, cloud source repositories, GitHub, or Bitbucket, and it executes a build to your specification. It produces artifacts such as binaries and Docker images. The Build config file contains the instructions for the cloud build to perform a task based on a specification. For example, your build config contains a function to build, package, and push Docker images.
* *Deployment automation*: GCP Code Deploy performs deployment automation. It deploys on any of the instances, including EC2 and on-premises server.
* *Unified CI/CD projects*: GCP Cloud code quickly develops, builds, and deploys the application on GCP. It provides a user interface to visualize and manage software development activities.

**DevOps Transformation**

The ultimate goal of DevOps is to unify development operations from end to end, but many organizations struggle to realize the full adoption journey from one application to the enterprise level. Challenges vary at every stage. Thus, even the most promising efforts fail to scale products and services through the entire scope of adoption.

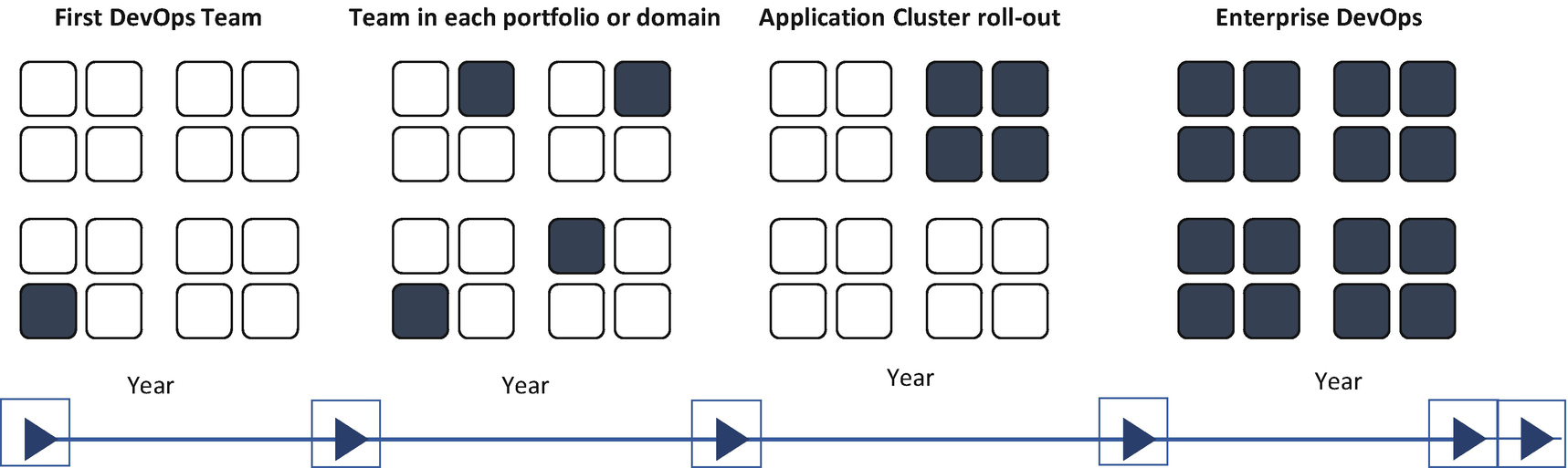
A DevOps journey is an organization-wide journey across all layers. Even if your scope of DevOps adoption is within a single layer, you need to sync with other layers. Individual applications are the basic consideration in your DevOps strategy. Stakeholders should keep enterprise adoption in mind when deciding the process, tools, and practices to implement.

The core tenant of DevOps is to identify dependencies among related applications and group them by release time and strategy. These groupings are known as *clusters*. This allows for the harmonious implementation of DevOps across all applications.

These are the key factors of a DevOps transformation:

* It’s a journey about reinventing yourself.
* Focus on people, process, and tools adoption.
* Learn from other teams and use their best practices.
* Start with the most valued product (MVP).
* Measure all the KPIs.

You are at the beginning of the enterprise DevOps transformation. As shown in Figure [14-14](https://learning.oreilly.com/library/view/cloud-native-architecture/9781484272268/html/511610_1_En_14_Chapter.xhtml#Fig14), DevOps is the logical next step in your agility.



***Figure 14-14***

DevOps transformation journey

The success of your DevOps transformation is based on how you use these perspectives:

* *IT landscape*: This includes practices and principles to build and configure your solution stack to enable autonomous, fast, and reliable software delivery.
* *Organization*: This includes tribe and team topologies, partnerships, a culture, and a skillset that encourage thinking across silos and enable tribes to become autonomous.
* *Practices*: How you work is a key capability that supports doing the right things in the right way.
* *Enabling practices*: This includes continuous automation testing, CI, CD, continuous deployment, monitoring.

The following are the challenges of DevOps transformation:

* *Governance*: Creating a governance framework that is effective at the speed of DevOps is a major hurdle for enterprises.
* *Product/project management*: The majority of software applications are still hosted on-premises, and support applications are dispersed across fragmented teams, business units, and organizations, leading to a lack of ownership.
* *Quality*: DevOps practices require continuous quality across the lifecycle.
* *Compliance*: Depending on the nature of the industry you are in, your enterprises need to adhere to various compliances like GDPR, SEC, etc.

To overcome these challenges in transformation, you need to adopt the following considerations across enterprises:

* *Transformation alignment*: Infuse DevOps during the transformation. This streamlines and reduces the overall governance issues.
* *Cultural and change management*: Organizations that ignore cultural and change management during the transformation journey fail to transform successfully. Like any other transformation, DevOps requires training mentorship, resource skilling, behavioral change, and motivational or reward programs.
* *Stakeholder management*: Every project or product has multiple stakeholders, but when a team fails to collaborate with stakeholders cohesively, adoption tends to fail.
* *Prioritizing application*: Some applications benefit more from DevOps than others. Selecting and prioritizing the right one is important for transformation success.
* *Tools setup and process design*: Tools and processes should be established before implementation.
* *Minimum viable product (MVP)*: Identify and create an MVP. This philosophy aims to provide early benefits and assurance to stakeholders before they invest in fully scaling on a DevOps transformation.
* *Identify clusters*: All dependent applications should move through development and testing cycles together in the same space.
* *Create a consolidated implementation plan*: Planning the consolidated release of an application cluster requires an assessment of the challenges at hand as well as the techniques to overcome them.
* *Organization structure*: Well-structured IT teams greatly enhance DevOps adoptions.

**Summary**

Automation is about combining agile, DevSecOps, SRE, and the cloud to build an elastic, hyper-speed organization. These four elements are organized around each other in the following ways:

* Talents and teams are like liquid and flow quickly.
* These four elements such as agile, DevSecOps, SRE and cloud are structured around intelligent software engineering.
* Full-stack teams are end-to-end accountable for projects.
* Applications are independent of each other.
* The 12-factor app allows automation.
* These four elements focus on resilience and failure tolerance.
* These four elements are use automated change management and resilience design patterns.
* These four elements are elastic and create a highly scalable infrastructure.
* These four elements have an immutable infrastructure with a self-service paradigm.

Following best practices, you must consider the following for your DevSecOps transformation:

* The journey is about reinventing yourself; do not restrict yourself from learning new things.
* Focus on people and interaction over tools and process.
* Keep sharing; learn from other teams.
* Prepare yourself to kickstart DevOps.
* Theory is good for understanding; try to realize it in projects.
* Train your team; DevOps is not for everyone.
* Focus is important; don’t mix too many user stories.
* Measure as much as possible.

In this chapter, I explained the basics of the DevOps pipeline and the security in it and also covered DevDataOps and DevNetOps and how they will help you to address data analytics and networking. These pipelines are an integral part of your cloud native journey. I also covered how to drive DevOps transformation to an enterprise.