Prodapt

Chase
Extraordinary

Magic of containerization

Modernize .NET apps and rapidly deliver new features to your customers

Credits | Sivasubramanian V Roshan Mehaboob Rohit Karthikeyan

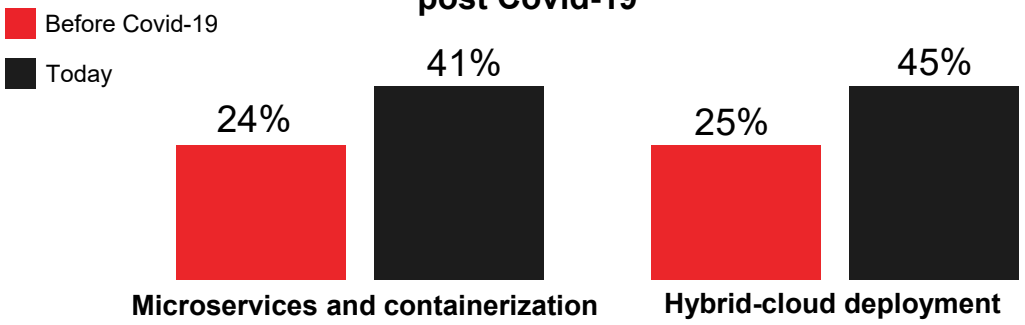
Current state of **IT infrastructure in the connectedness** industry

- **>70% of servers around the globe use Windows** Operating System (OS), according to Statista report
- Majority of the Windows workloads **involve .NET-based legacy applications**
- These .NET-based legacy apps are **not cloud-native and are running on-premises**
  - E.g. of .NET-based apps: **credit check, billing, order management**, etc

Urgent needs of CIOs

- Modernize .NET apps, to **achieve scalability, and improve performance**
- Reduce **OS licensing footprint by containerizing** .NET apps
- Transform into a **hybrid or cloud-native** company

Investments in legacy app modernization services pre & post Covid-19



App modernization services have **witnessed a growth of more than 40%** as service providers shift to cloud-first solutions

Source: HFS Research

Challenges faced by CIOs & technology decision makers to modernize .NET-based legacy apps

.NET versions prior to **Windows Server 2016 do not support containerization**

Scalability issues of .NET legacy apps on-premises

Containerization **wasn't created with Windows** in mind. It was built with Unix in mind.

Re-writing all the .NET apps for the latest Windows version is a time-consuming process

Migrating .NET apps fully to **cloud is expensive**

Lack of clear migration strategies of .NET apps leads to lot of re-engineering efforts

Prodapt Chase Extraordinary

Options to help **overcome the challenges of .NET app modernization**

CIOs and technology decision makers must decide what to do with the .NET legacy apps

Option 1

- Maintain the status quo and do nothing

→ Leads to app inefficiency, security risks, and incompatibility with modern technologies

Option 2

- Rearchitect all .NET apps to run as containerized workloads

→ Hard and time-consuming efforts

Option 3

- Refactor to **.NET Core & containerize**
- **Modernize** .NET apps & share workloads across cloud

→ Achieve significant cost savings, accelerate modernization, & achieve seamless migration

Adopting a well-defined modernization strategy with containerization, will transform .NET apps to be cloud-ready and empower service providers to build and run scalable applications

Key transformation levers to **successfully containerize and modernize** .NET-based legacy applications

3-step process levers to containerize .NET apps and share workloads across cloud

Performance optimization with CI/CD pipeline and observability platform

1

Systematic approach to assess if a .NET-based legacy app can be containerized

2

Platform to containerize and deploy .NET-based **legacy applications on premises**

(Note:** This lever is recommended for service providers who want to remain on-prem or want to be closer to their data centre, or in need to share workloads across hybrid environment**)

3

Platform to containerize and **deploy** .NET-based **legacy apps across hybrid cloud** environment

4

A secure app delivery platform with CI/CD to **achieve fully automated zero-touch pipelines**

5

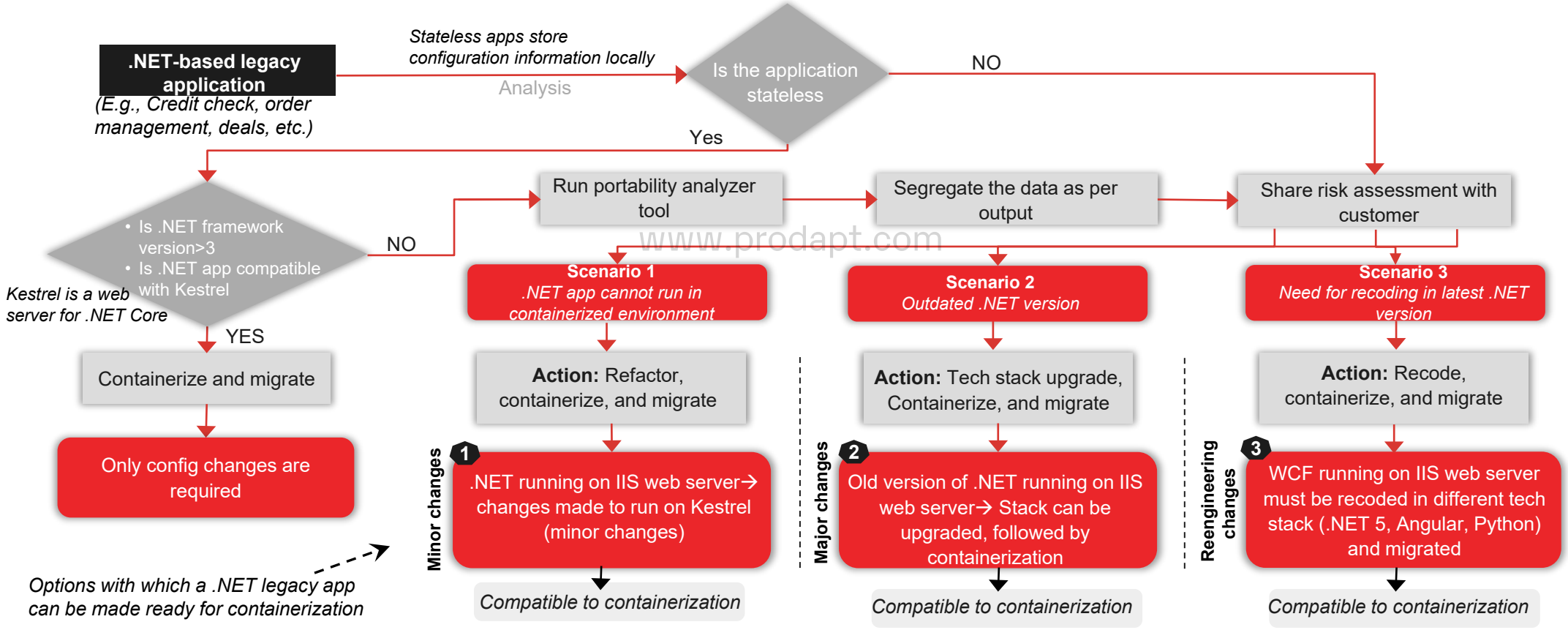
Multi-cluster observability platform to achieve single source of truth, for monitoring .NET apps

By embracing these transformation levers, service providers can ensure a successful containerization and modernization of their .NET-based legacy applications, attaining a **2.5X reduction in feature delivery time**, and improve application **performance by 40%**

Systematic approach to assess if a .NET-based legacy app can be containerized

1 2 3 4 5
A B

Containerization compatibility evaluation framework



The flowchart helps to check if a.NET-based legacy app is ready for containerization or not

Prodapt Chase Extraordinary

Systematic approach to assess if a .NET-based legacy app can be containerized

1 2 3 4 5
A B

Challenge

- In traditional methods, service providers do not follow a systematic approach for assessing a .NET app's compatibility to containerization
- Migrating .NET apps without checking its compatibility will lead to app scalability issues in the future

Recommendations

- Rewrite the apps which are stateful, to leverage persistent volumes. A stateless app is dependent on third-party storage because it doesn't store any kind of state in memory or disk.
- Check if the .NET app is compatible with Kestrel, an open-source web server, used to host the apps on any platform
- Use .NET Portability Analyzer tool to scan through the source codes and determine if any APIs are missing or libraries to be ported on specified targeted .NET platforms
- Segregate the output data, to check if the Dynamic-link Libraries (DLLs) are compatible to containerization
- Perform risk assessment and move the .NET app into a separate database, if the app is stateless. Recode the app in a different tech stack (.NET 5.0, Angular, Python) to make it compatible to containerization

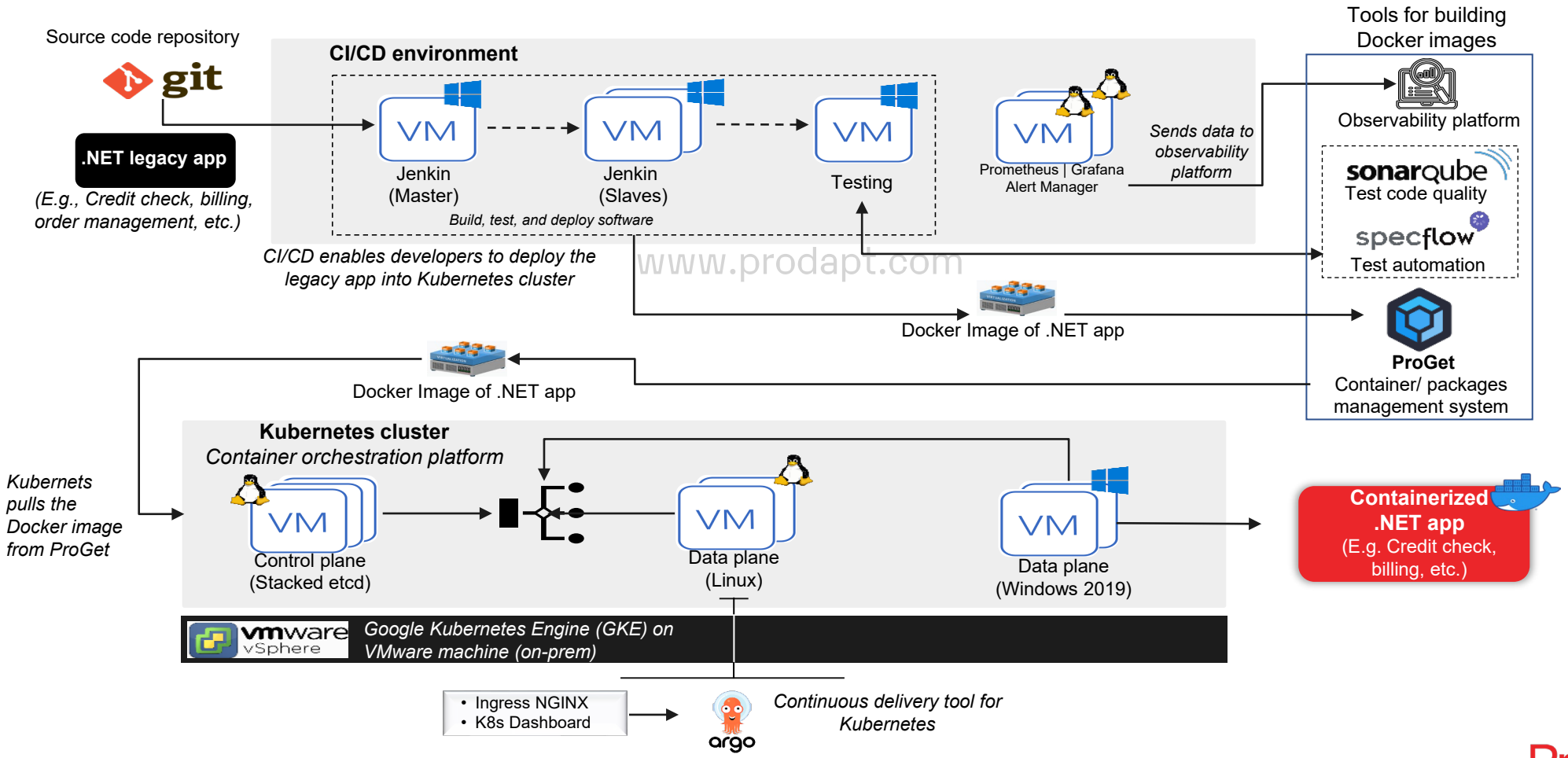
Benefits

- Efficient use of compute resources
- Containerized .NET apps scale based on usage
- A proper evaluation of the legacy app will help service providers to compile and create a containerized package in a shorter span of time
- The end-to-end deployment of an application can be achieved in less than 15 mins
- Isolated apps improve security posture
- Easily build and tear down environment based on need, using orchestration tools like Kubernetes

Platform to containerize and deploy .NET-based **legacy applications on premises**

1 2 3 4 5
A B

Reference architecture to containerize .NET legacy apps on-prem



Prodapt Chase Extraordinary

Platform to containerize and deploy .NET-based **legacy applications on premises**

1 2 3 4 5
A B

Recommendations

- Use Jenkins, an open-source automation server, to automate building, testing, and deploying of application in the CI/CD environment. Jenkins is used to create the Docker image (for containerization) and push it to ProGet.
- Use Google Kubernetes Engine (GKE) to deploy, orchestrate, and manage the Docker images in the Kubernetes pods. A pod is a collection of one or more containers and is the smallest unit of a Kubernetes application.
- Leverage a container management tool, such as ProGet, to store and manage the different versions of Docker images
- Use multiple environments with Infrastructure as Code (IaC). With IaC and ArgoCd, new environments can be created in a click of a button. This improves productivity and decreases testing lead time.
- Use SonarQube, an open-source platform, to test the quality of codes developed in CI/CD environment
- Use SpecFlow, a test automation tool, to check if the .NET application has met the required standards

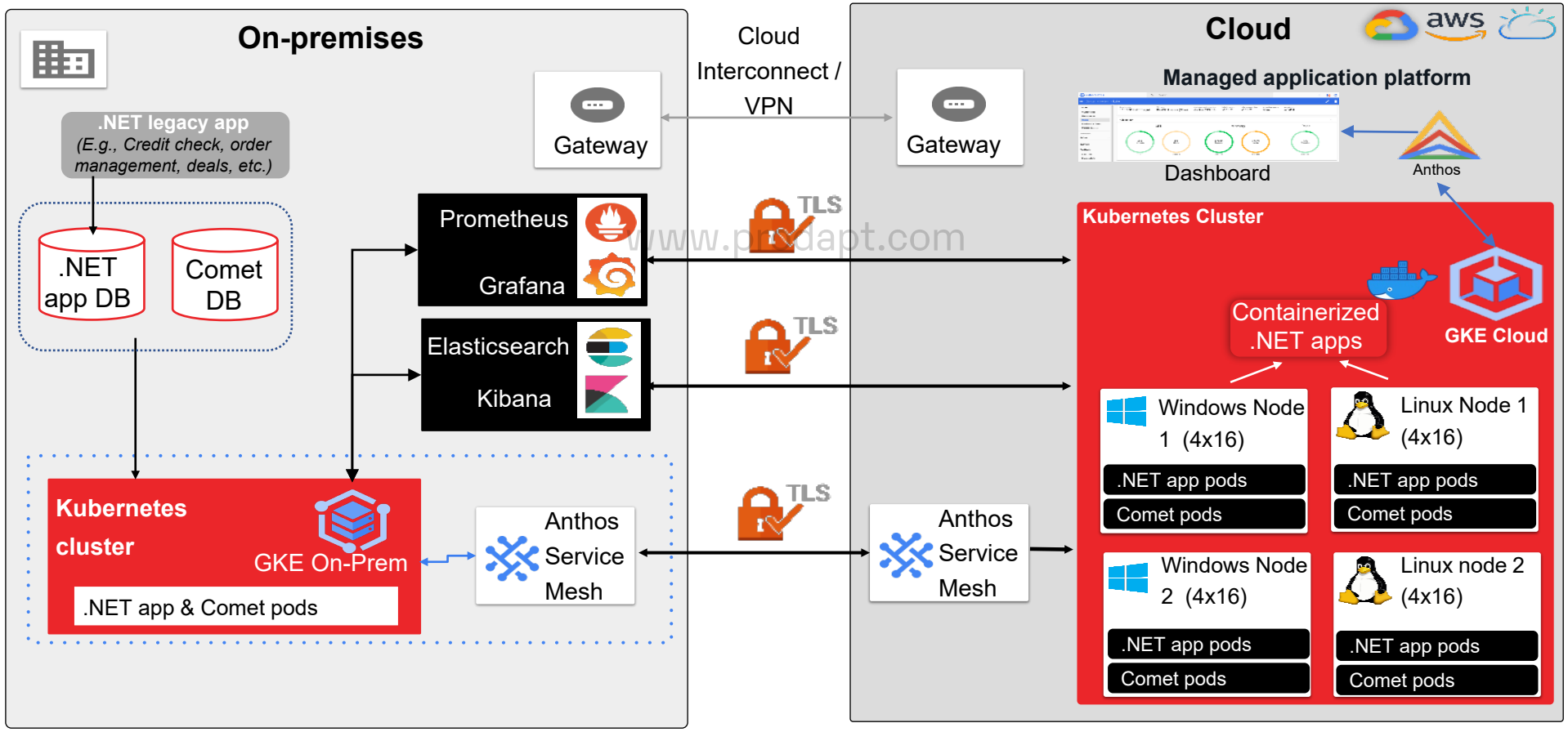
Benefits

- Existing infrastructure (on-prem) can be leveraged to run Kubernetes workloads
- With Google support tools, upgrades and security patches can be implemented without hassles or downtime.
- 3X improvement of release frequency into production
- GKE accelerates the process of .NET app containerization

Platform to containerize and deploy .NET-based **legacy apps across hybrid cloud environment**

1 2 3 A B 4 5

Reference architecture to containerize .NET-based legacy apps across hybrid environment



Prodapt Chase Extraordinary

Platform to containerize and deploy .NET-based **legacy apps across hybrid cloud environment**

1 2 3 4 5
A B

Recommendations

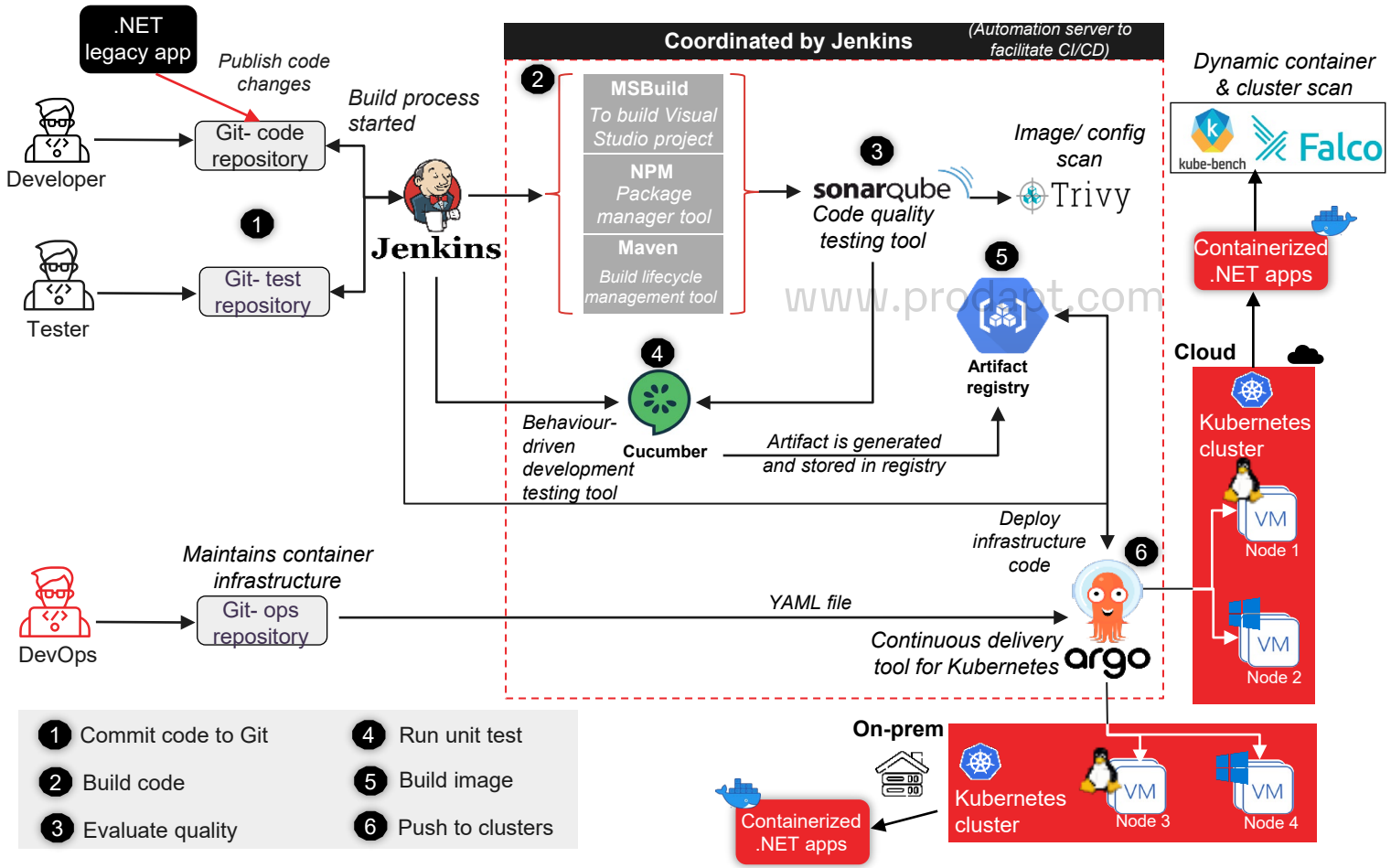
- Use Anthos to manage GKE clusters and workloads running on virtual machines across hybrid environments
- Leverage cloud's compute power to expand workload (Pods) requirements in on-prem Kubernetes cluster. This helps to manage the additional workloads on premises.
- Push the Docker image to Google Container Registry using gcloud, so that the image can be later referred when the Kubernetes cluster is deployed
- Leverage Anthos Service Mesh to manage traffic between services while monitoring, troubleshooting, and improving application performance
- Use Prometheus, an open-source monitoring tool, to monitor the Kubernetes cluster in both on-prem and cloud
- Use Cloud Interconnect tool to extend on-prem network to cloud network through a low latency connection
- Use Elasticsearch tool for analytics and operational intelligence use cases

Benefits

- Flexibility to share workloads across on-premises and cloud environment
- Reduce feature delivery time by 2.5X
- Reduce Operating System (OS) licensing footprint by containerizing more applications in single server

A secure app delivery platform with CI/CD to **achieve fully automated zero-touch pipelines**

Reference architecture for fully automated-zero touch pipelines



- ❑ The automated zero-touch pipeline is highly secure
- ❑ The DevSecOps ensures security right at the starting point of the development process. It enables the identification of vulnerabilities early and strengthens security throughout the application lifecycle

Cloud-native app delivery platform with CI/CD to **achieve fully automated zero-touch pipelines**

1 2 3 4 5
A B

Recommendations

- Use Argo CD, a controller tool, to continuously monitor all running applications and comparing their live state to the desired state
- Adopt trunk-based development practice to merge small, frequent updates to the main branch, thereby increasing the software delivery efficiency
- Use MSBuild plugin to build .NET and Visual Studio projects
- Use Maven tool to manage build dependencies and to work with plugins that allow users to add other tasks to the standard compile, test, package, and deploy tasks
- Leverage NPM (Node Package Manager) tool to manage multiple versions of code and code dependencies
- Adopt Artifact Registry to manage container images and language packages
- Deploy multiple containers in a container group by specifying containers configuration in a YAML configuration file

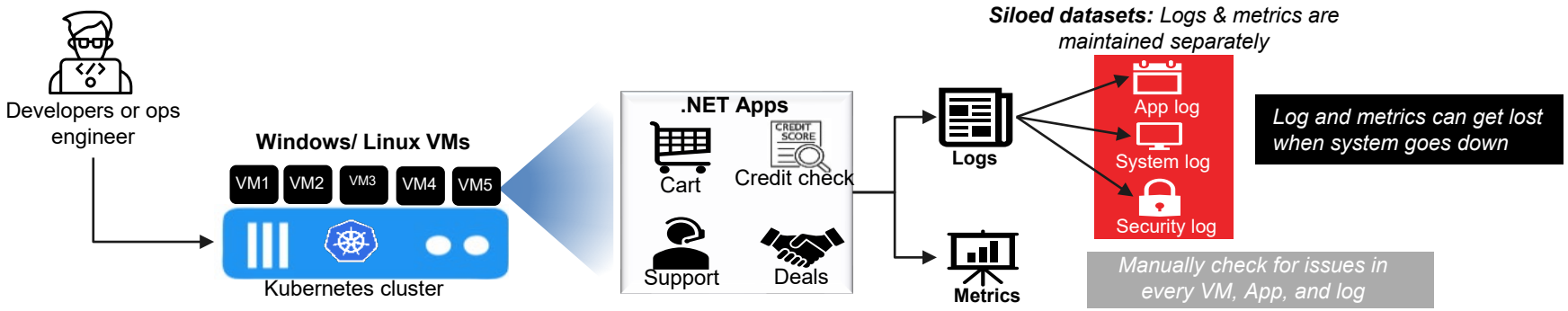
Benefits

- Reduce time to market of new features
- Improve application performance by 40%

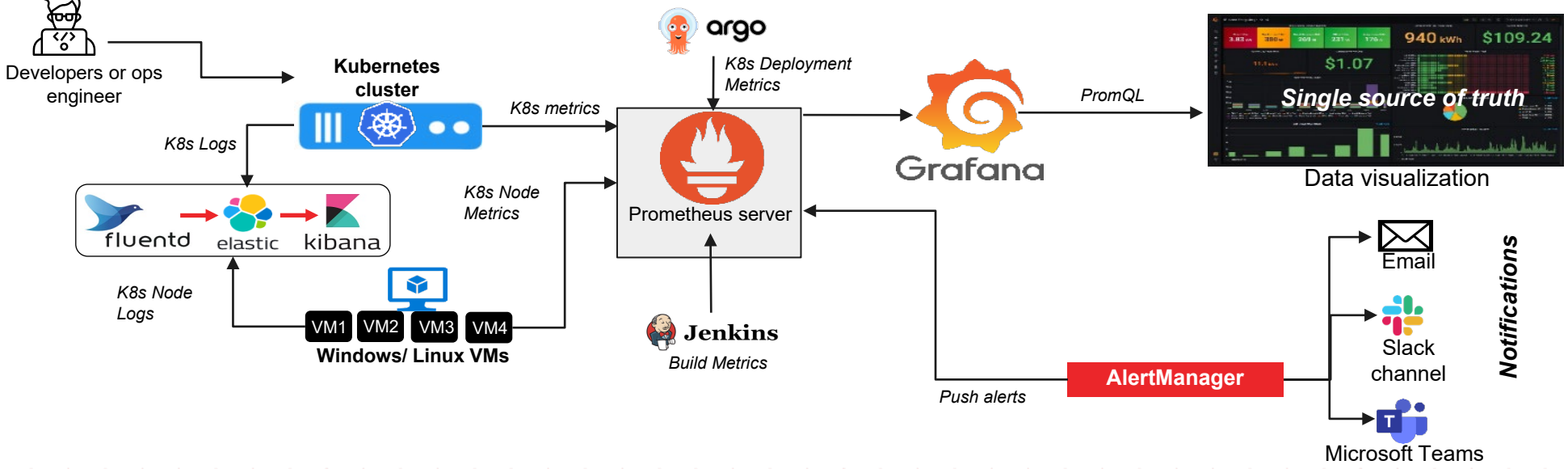
Multi-cluster observability platform to achieve single source of truth, for monitoring .NET apps

1 2 3 4 5 A B

Traditional approach to manually monitor infrastructure and .NET apps



Modernized approach with observability platform, to monitor infrastructure and .NET apps



Prodapt Chase Extraordinary

Multi-cluster observability platform to achieve single source of truth, for monitoring .NET apps

Challenge

- There is a need for manual intervention to monitor the VM and the .NET apps for any issues. E.g., if there is infrastructure issue, the respective logs and metrics must be checked to identify the issue type.
- As there are multiple VMs, there is a risk for some VMs to crash. Hence there is a need for the logs and metrics to be maintained separately in different machines.
- If the VMs, in which the logs and metrics are maintained gets crashed, then there is a risk of data getting lost.

Recommendations

- Enable monitoring with Prometheus, Grafana, and AlertManager stack, to deliver cluster-level metrics and machine-level metrics
- Use Prometheus, a time series database, that works by polling metrics endpoints and processing the data exposed by endpoints
- Leverage Fluentd, an open-source data collector, to unify the data collection and consumption for a better use and understanding of data
- Use Elasticsearch tool for analytics and operational intelligence use cases
- Use Ingress metrics to alert on failures, and proactively investigate the app issues
- Leverage Kibana, an open user interface, which lets to visualize the Elasticsearch data and navigate the Elastic Stack
- Use Grafana, a data visualization and analytics tool, to build dashboards and graphs for log and metrics data
- Deploy AlertManager to handle alerts generated by Prometheus and route them to integrations like email notifications

Benefits

- The observability platform removes data silos and provides a single pane of glass view of data
- The platform provides instant insights into the real-time performance of Kubernetes clusters and .NET apps, which enables to take timely action
- Software tools such as Prometheus, Grafana, Elasticsearch, etc. are open-source and CNCF-approved, making them inexpensive and easy to set-up

Benefits achieved by a leading service provider in Europe after modernizing their .NET-based legacy applications

Implementing the key transformation levers delivered the following benefits

2.5X
reduction in feature delivery time

3X
improvement in release frequency of applications

<15 mins
reduction in time to build & deploy apps

40%
improvement in application performance

No manual intervention required

Improved security posture & observability of application

Reduced time-to-market of new features

THANKS!