ISync

# Jenkins as a Service (JaaS)

A Road to manage your finances better

Rohit Agarwal
Product Owner

Experienced IT program manager specializing in infrastructure, security, and SRE management.

Meng Ni
Scrum Master

Skilled in agile coordination and platform efficiency improvement.

Kush Shah
Quality Assurance/Tester

Experienced in software testing and risk management for secure IT systems.

Mutian He
Software Developer

Focused on developing and integrating applications in cloud and PaaS environments.

Het Bhavesh Shah
Solution Architect

Expert in PaaS integration, enterprise security, and resilient cloud design.

ISync

# TABLE OF CONTENTS

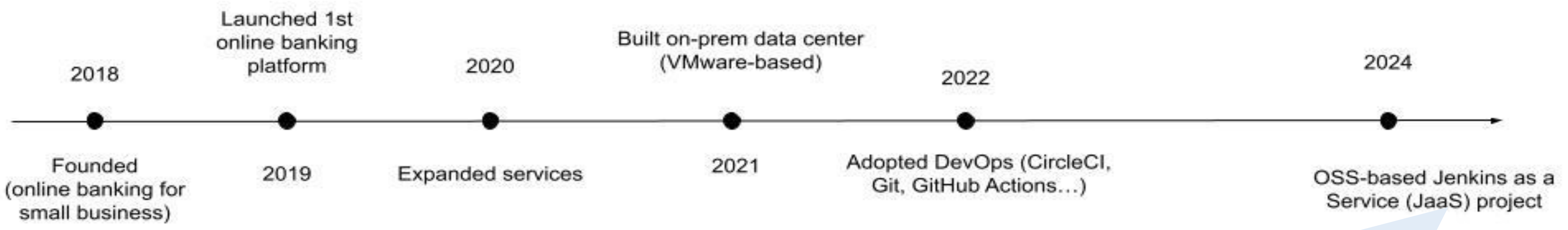
BACKUP

ISync

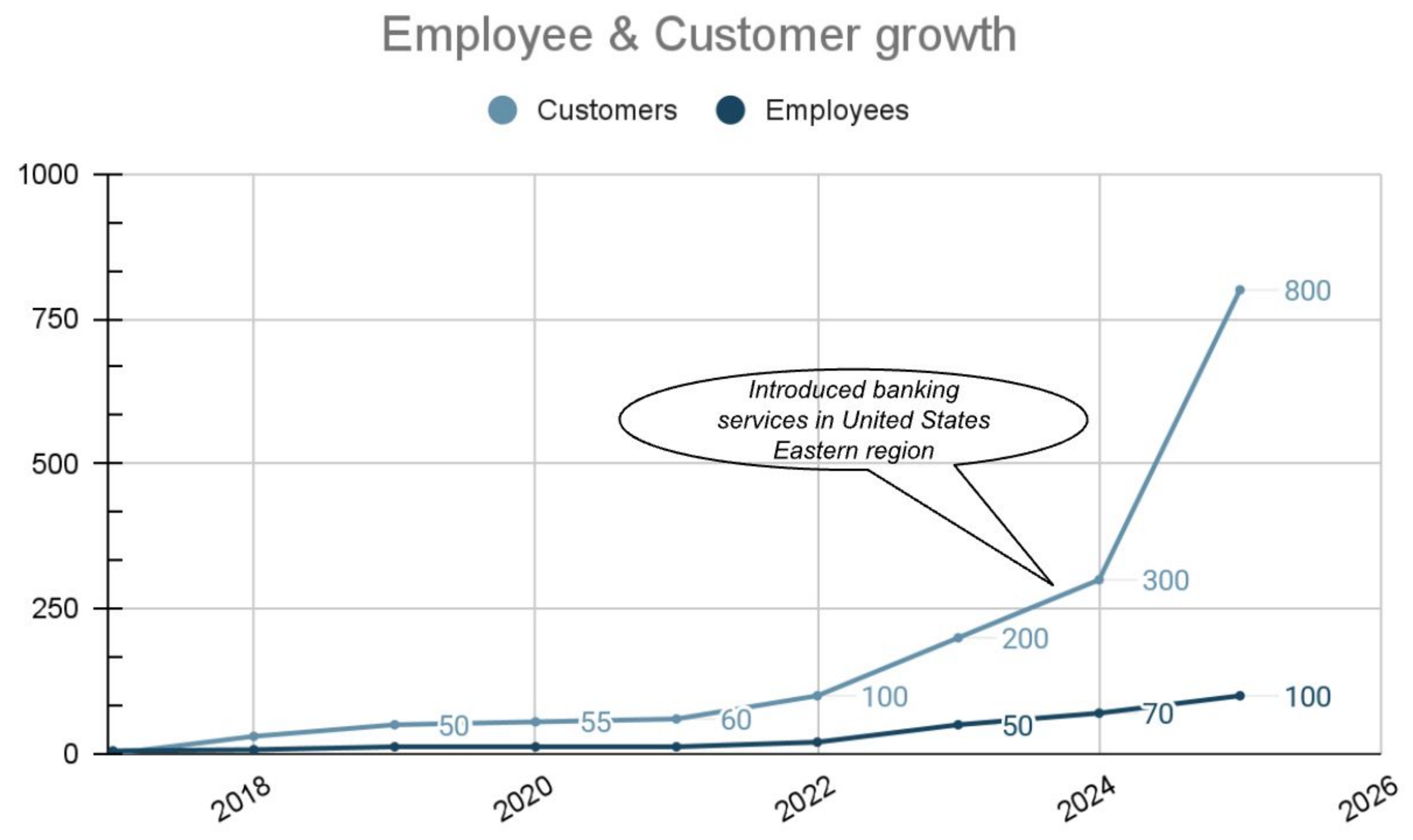
# PROJECT SUMMARY

ISync

# PROJECT SCOPE AND CONTEXT


2018
Founded (online banking for small business)
Launched 1st online banking platform
2019
2020
Expanded services
Built on-prem data center (VMware-based)
2021
2022
Adopted DevOps (CircleCI, Git, GitHub Actions…)
2024
OSS-based Jenkins as a Service (JaaS) project

ISync aims to standardize its fragmented CI/CD ecosystem by adopting Jenkins as the unified enterprise solution, which helps us consolidate build and deployment pipelines under a centralized environment.

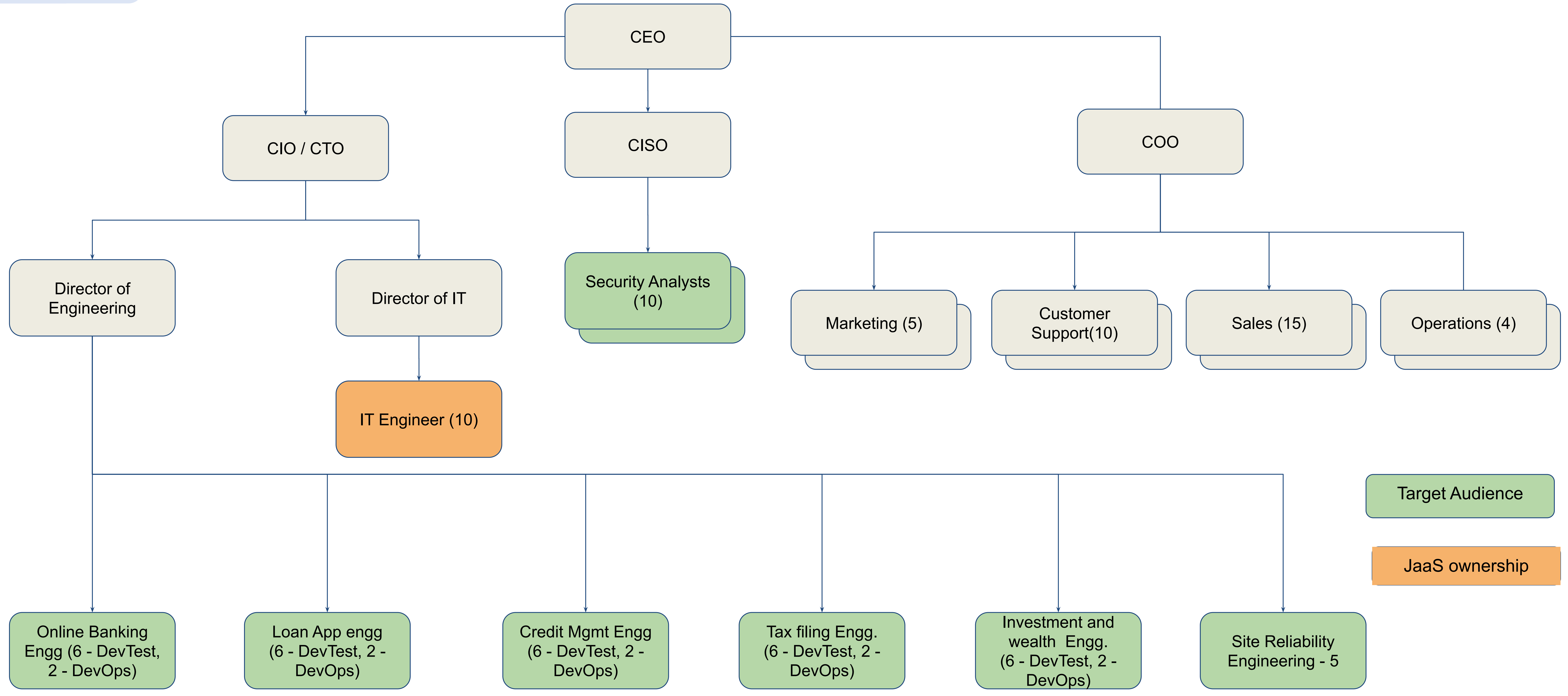


- **Company Name:** FinTech Solutions Inc.
- **Industry:** Financial
- **Company Size:** ~ 100 employees
- **Headquarters:** Santa Clara, California
- **Core Services:** Online banking, Investment and wealth management, tax filing, loan and credit management services
- **Customers' footprint**: United States of America

SIMPLE

ISync

# ORGANIZATION STRUCTURE



CEO
CIO / CTO
CISO
COO
Director of Engineering
Director of IT
Security Analysts (10)
Marketing (5)
Customer Support(10)
Sales (15)
Operations (4)
IT Engineer (10)
Target Audience
JaaS ownership
Online Banking Engg (6 - DevTest, 2 - DevOps)
Loan App engg (6 - DevTest, 2 - DevOps)
Credit Mgmt Engg (6 - DevTest, 2 - DevOps)
Tax filing Engg. (6 - DevTest, 2 - DevOps)
Investment and wealth Engg. (6 - DevTest, 2 - DevOps)
Site Reliability Engineering - 5

CircleCI, Git, Selenium

GitHub Actions, Git, Selenium

GitLab, Perforce, Cypress

CircleCI, Perforce, Selenium

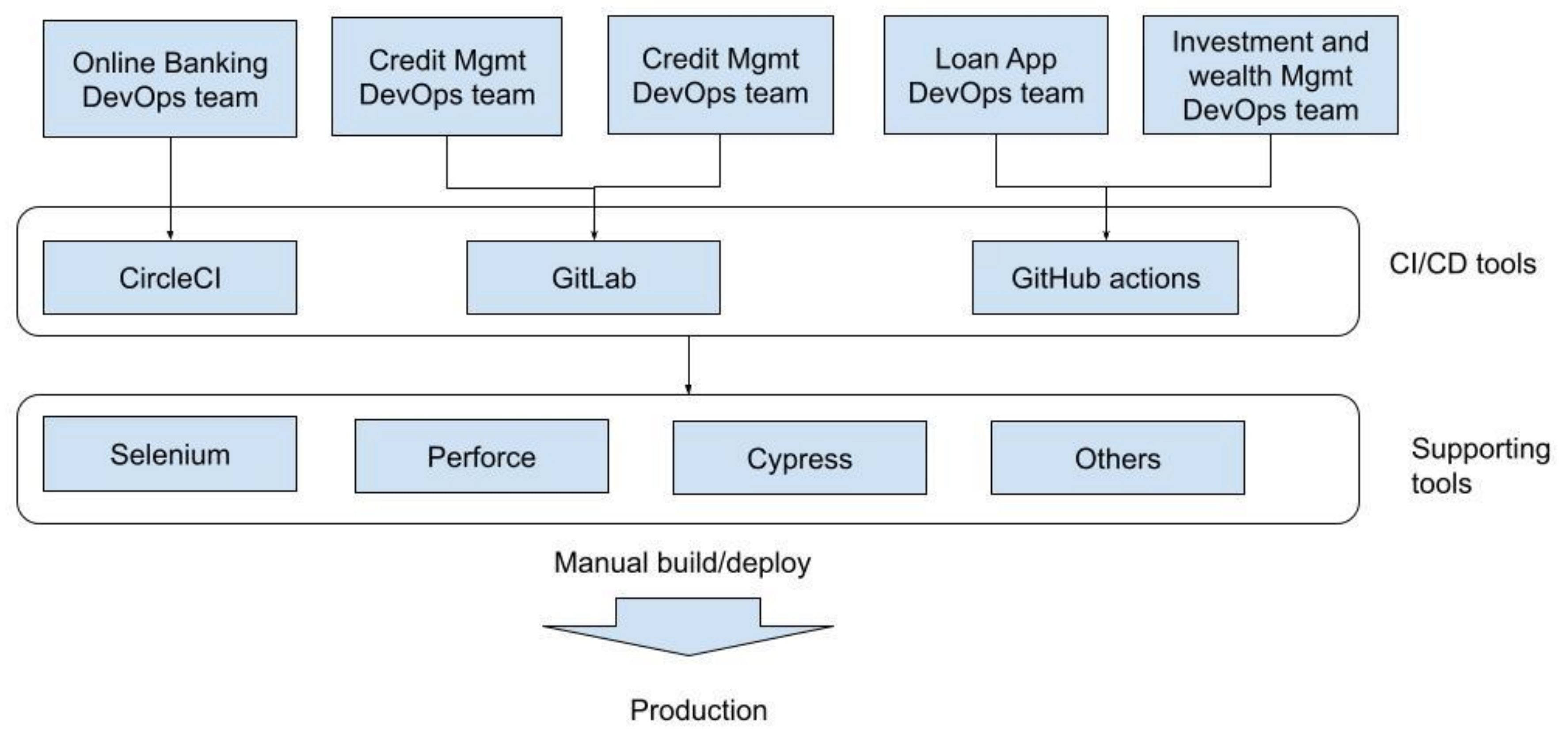
GitLab, Git, Selenium

GitLab, GitHub Actions, CircleCI, Perforce, Cypress, Selenium

SIMPLE

ISync

## PROBLEM STATEMENT



**Pain points**

1. Tool Fragmentation and Lack of Standardization [1]
   Different CI/CD tools use different configuration formats and plugins, making it hard to maintain consistency.
2. Testing & Integration Challenges [1]
   There is limited visibility into testing coverage, and cross-module integrations often fail due to incompatible setups.
3. High Licensing and Maintenance Costs
   Each CI/CD tool has its own subscription and maintenance fees, increasing overall expenses.
4. Compliance and governance risks [2]
   No unified access control or audit policy may leads to potential compliance gaps.

### Current State

Each DevOps team operates independently with its own CI/CD tool.

- There is no unified automation process, resulting in manual build and deployment across all environments.
- The current setup lacks centralized monitoring, audit logging, and deployment rollback capabilities.
- Testing frameworks are inconsistent(Selenium vs. Cypress) making cross-team collaboration difficult.

### Future State

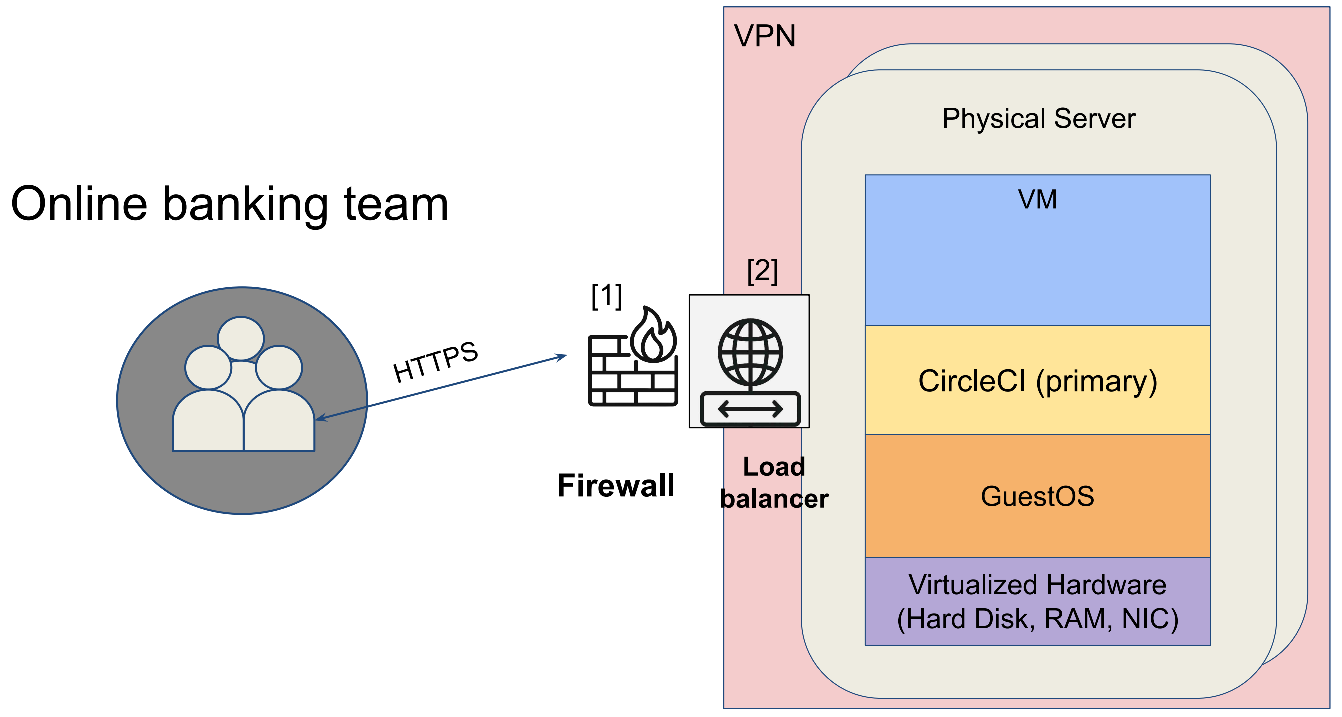
All DevOps teams will use a single CI/CD platform (Jenkins) with standardized pipeline templates.

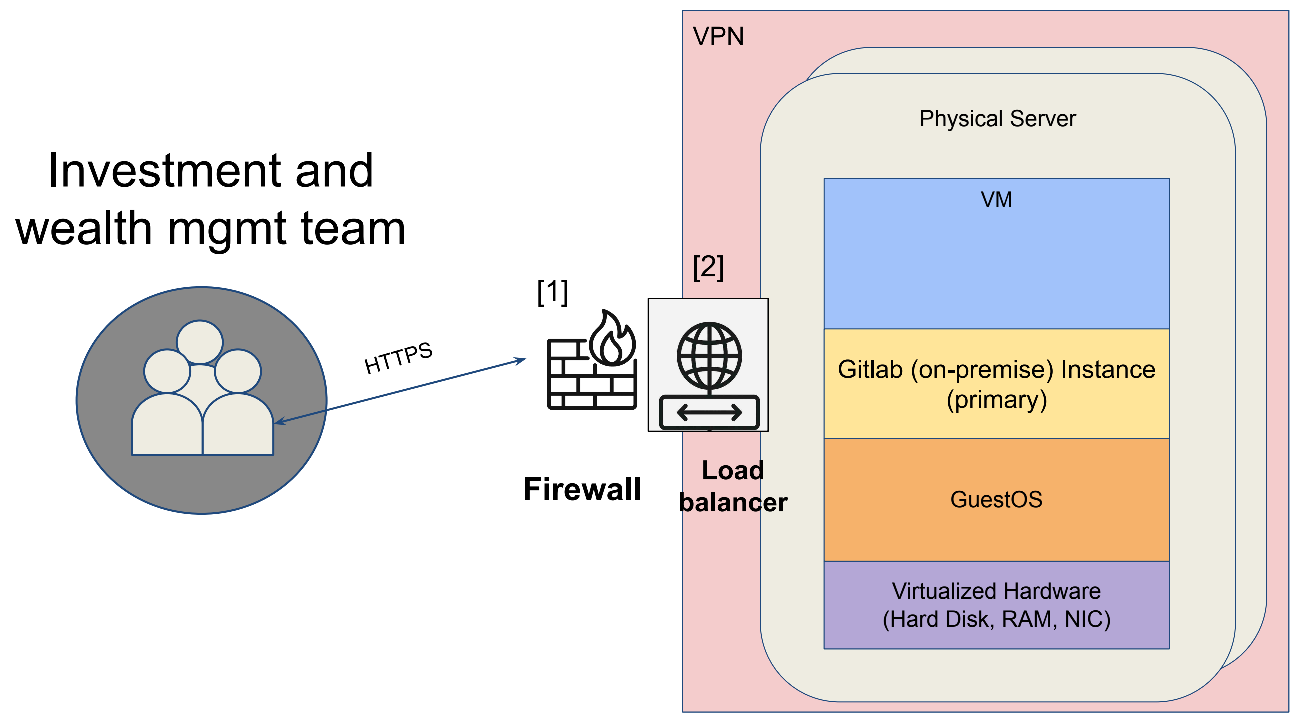
- The build, test, and deployment processes will be fully automated, replacing manual steps.
- Security will be strengthened through centralized access control, SSO integration, and audit logs.
- The company will eliminate redundant tool subscriptions, reducing maintenance costs and improving efficiency.

[1] Awais Akram. *From Fragmentation to Focus: Conquering DevOps Tool Chaos*. Published on Medium, 2023. Retrieved from https://medium.com/@awais.akram11199/from-fragmentation-to-focus-conquering-devops-tool-chaos-7d14f2a45e2e
[2] Michael Khusid. *Automating Compliance in CI/CD Pipelines: A Modern Software Development Framework*. SSRN, 2024. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5139607

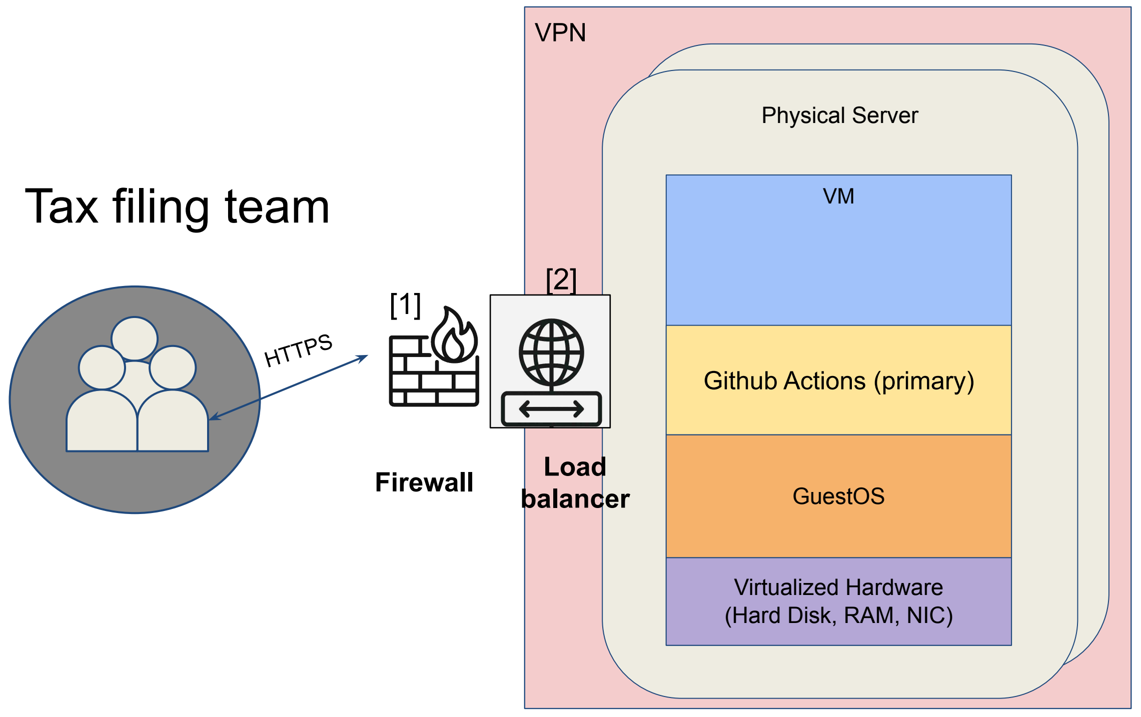
SIMPLE

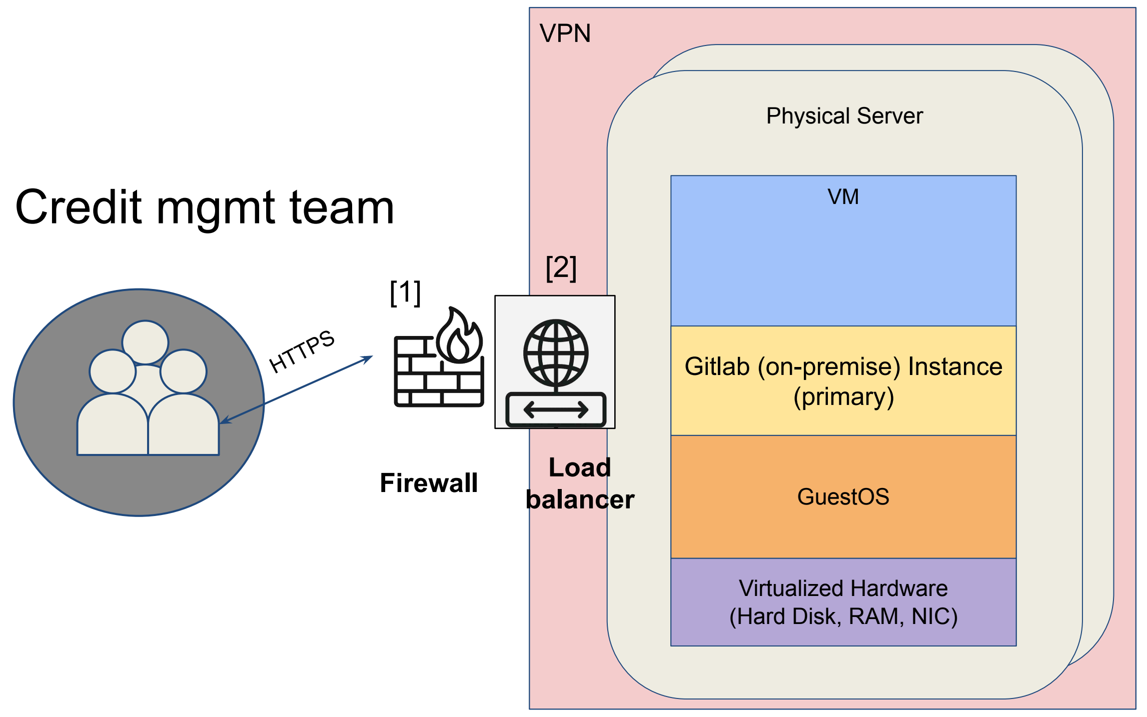
# AS-IS ARCHITECTURE (APP & INFRASTRUCTURE)

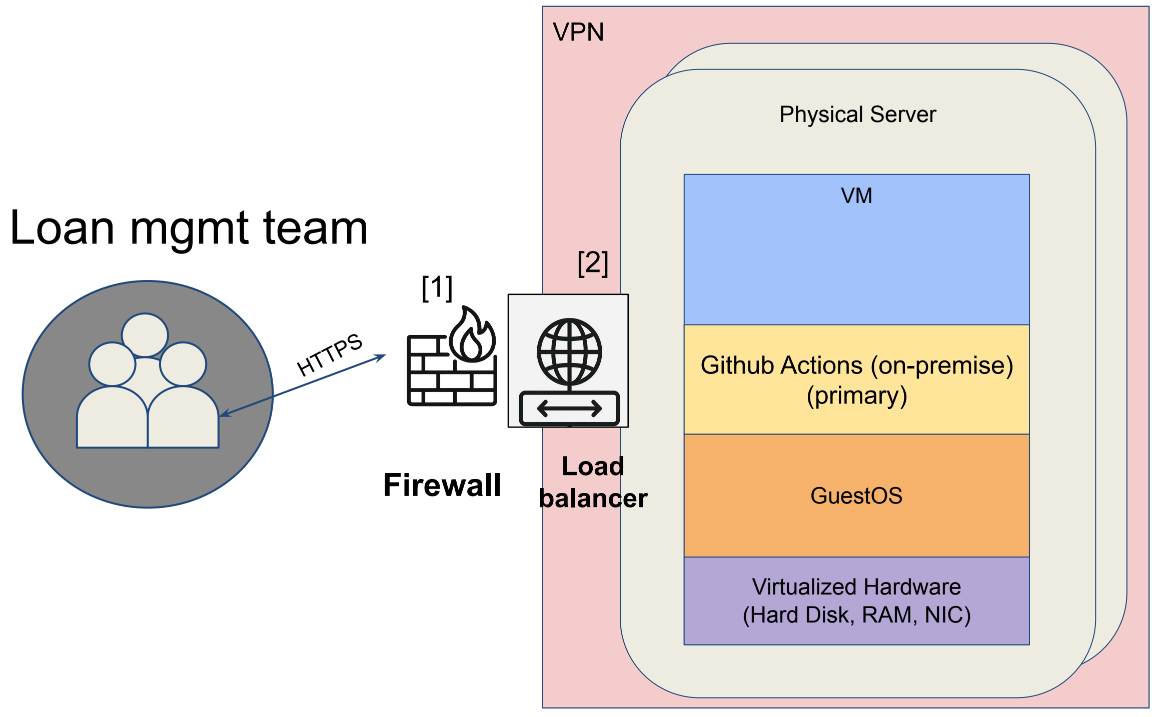



Investment and wealth mgmt team
HTTPS
[1]
Firewall
[2]
Load balancer
VPN
Physical Server
VM
Gitlab (on-premise) Instance (primary)
GuestOS
Virtualized Hardware (Hard Disk, RAM, NIC)

Each team runs their own instance of the respective CI/CD tool




Credit mgmt team
HTTPS
[1]
Firewall
[2]
Load balancer
VPN
Physical Server
VM
Gitlab (on-premise) Instance (primary)
GuestOS
Virtualized Hardware (Hard Disk, RAM, NIC)


Loan mgmt team
HTTPS
[1]
Firewall
[2]
Load balancer
VPN
Physical Server
VM
Github Actions (on-premise) (primary)
GuestOS
Virtualized Hardware (Hard Disk, RAM, NIC)

[1] firewall image generated with the help of Microsoft Copilot (SCU organization). Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).
[2] load balancer image generated, cropped to fit the requirement for the slide, using Microsoft Copilot (SCU organization) AI tool. Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).

AVERAGE

ISync

## VISION & SMART GOALS

**Our vision is to build a centralized Jenkins as a Service (JaaS) platform** on VMware that provides one standardized, secure, and automated CI/CD pipeline with rollback capability to streamline development, ensure compliance, and improve operational efficiency[1].

- **Increase development productivity**: Implement Jenkins-as-a-Service (JaaS) to reduce CI/CD pipeline setup time by **30 %** and enable standardized deployment across all teams **within several weeks**.
- **Cut operational costs**: Use one Jenkins platform instead of several CI/CD tools to **save around 30%** in annual license and maintenance costs by removing extra tools like CircleCI and GitHub action.
- **Increase release reliability**: Achieve **95 % successful release rate** through Jenkins automation with rollback and continuous monitoring within the project timeline.
- **Strengthen security & compliance**: Integrate Jenkins with VMware's centralized RBAC and audit logging to ensure **100 % compliance** with company security policies within the current project cycle.
- **Optimize infrastructure efficiency**: Host Jenkins on VMware to improve resource utilization and reduce server overhead by **15 %** compared to decentralized tools[2].

[1] TechMagic. *7 Reasons You Need DevOps as a Service ASAP.* Available at: https://www.techmagic.co/blog/x-reasons-you-need-devops-as-a-service
[2] Caepe. *How CI/CD Automation Saves You Time and Money.* Available at: https://caepe.sh/ci-cd-automation-saves-time-money/

SIMPLE

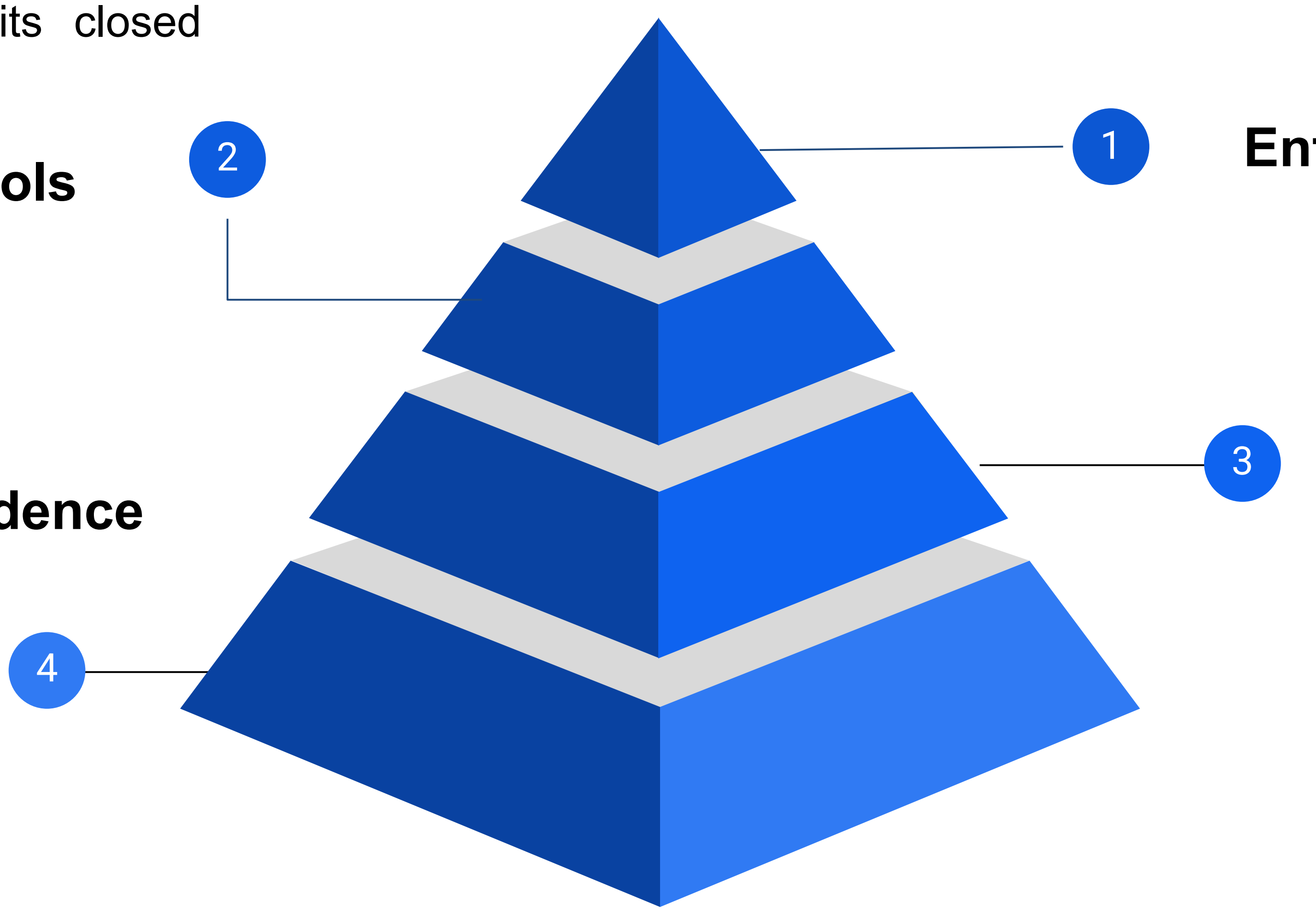
# WHY JENKINS[1]

- With 2000+ plugins[2] and support for every major build/test tool (Java, Go, Node.js, Python, etc.), Jenkins integrates seamlessly across hybrid and on-prem infrastructures (VMware, Docker, Kubernetes).
- CircleCI, while polished, offers limited extensibility and slower plugin innovation due to its closed ecosystem.

## Extensibility and support for DevTools

- Jenkins' self-hosted architecture gives ISync full control over data, security, and compliance—critical for regulated enterprise environments.
- Built on an MIT-licensed open core, ISync can modify, extend, and govern pipelines internally—something CircleCI managed SaaS model restricts.

## Enterprise-Grade Control & Governance



## Proven Reliability and Open Innovation

- Backed by the Continuous Delivery Foundation (CDF), Jenkins ensures transparent releases, LTS stability, and long-term community governance.
- CircleCI proprietary updates are managed centrally, giving less visibility and flexibility to enterprise users.

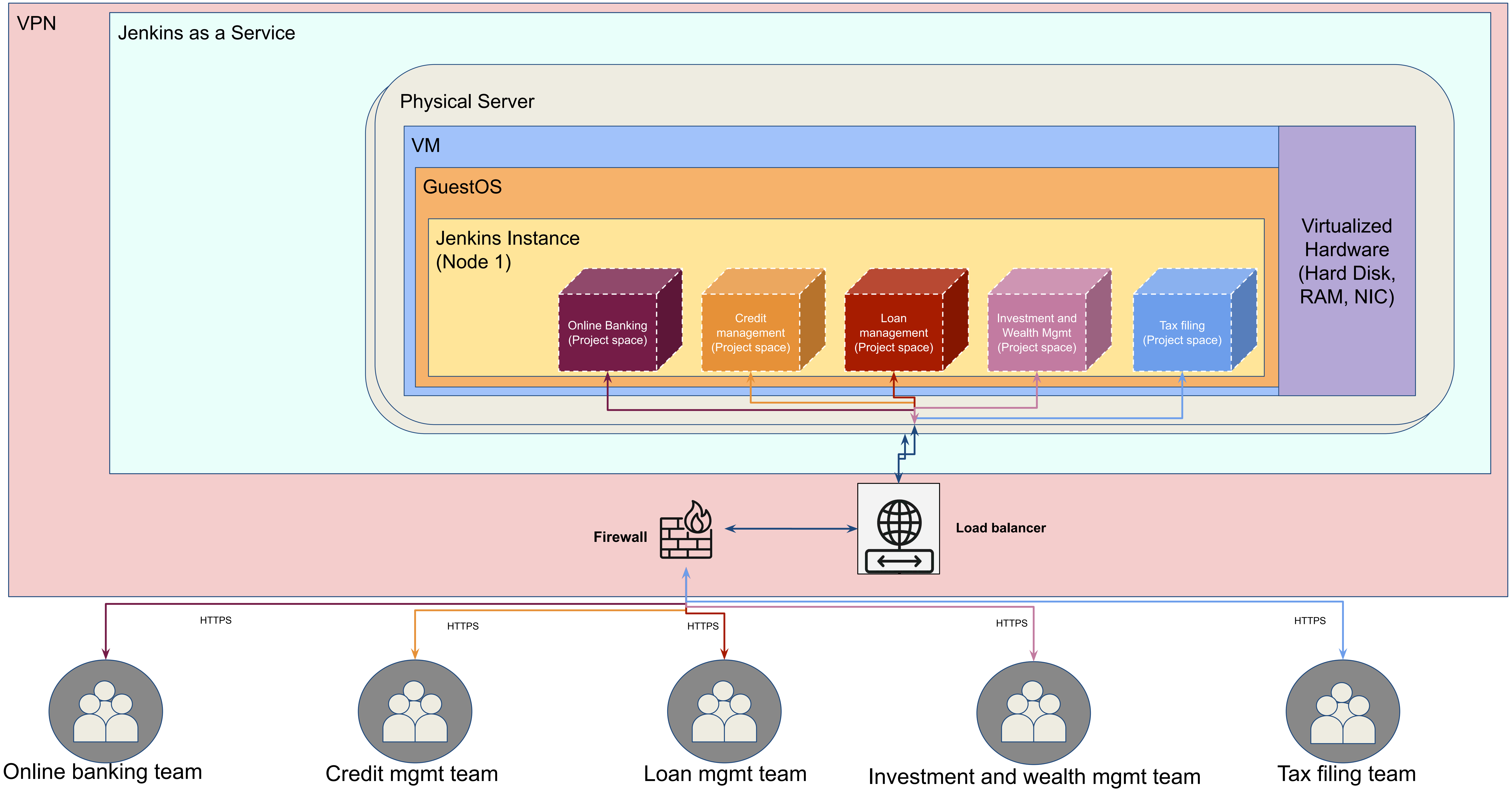
## Cost Efficiency and Vendor Independence

- Jenkins' open-source MIT model removes recurring license fees and vendor lock-in, enabling sustainable CI/CD scaling for ISync.
- CircleCI per-seat pricing and proprietary model can increase costs for large teams.

[1] Jenkins contribution reference: https://www.jenkins.io/blog/2019/05/30/becoming-contributor-newbie-tickets/
[2] Jenkins plugins marketplace: https://plugins.jenkins.io/

SIMPLE

# TO-BE ARCHITECTURE (APP & INFRASTRUCTURE)

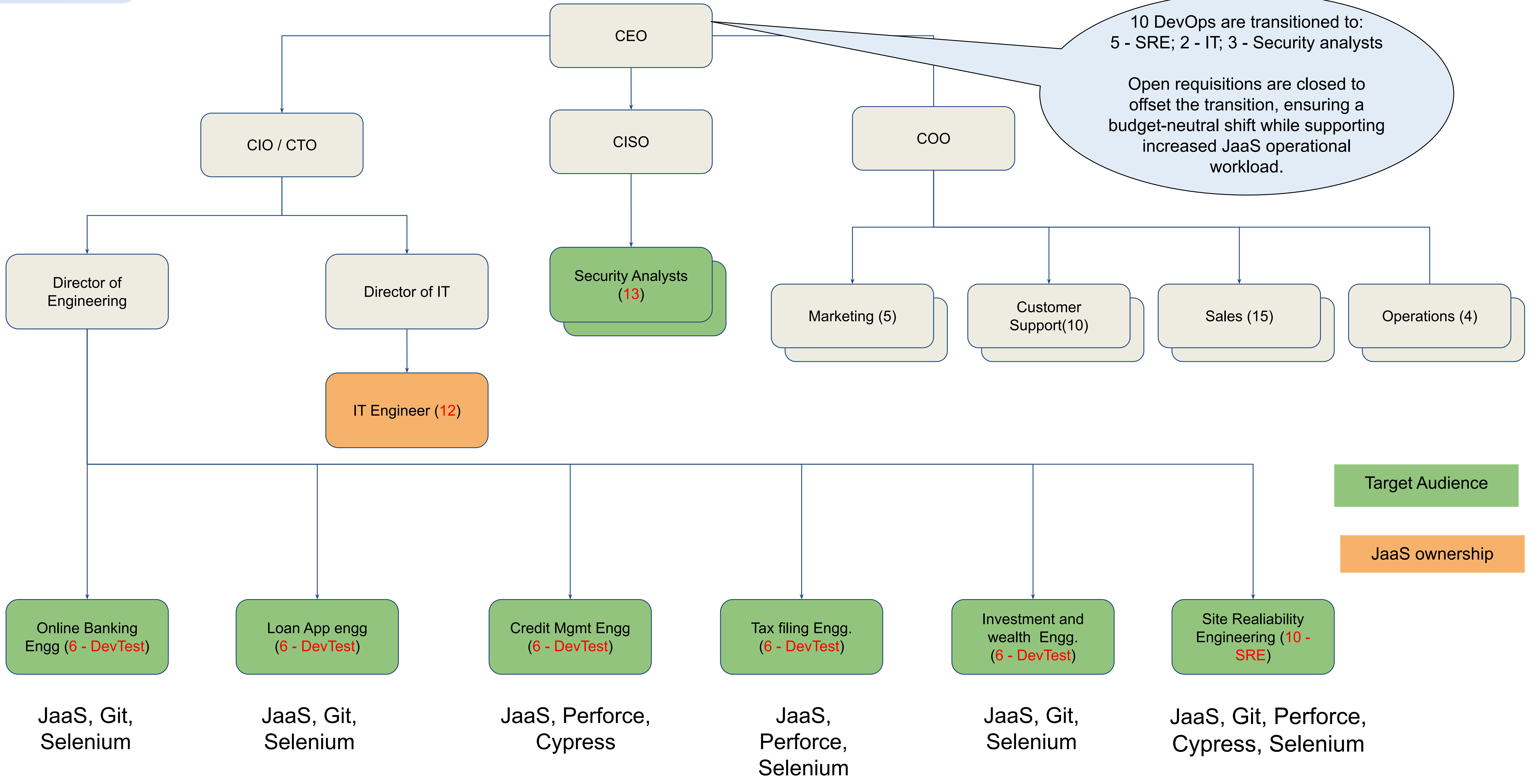


VPN
Jenkins as a Service
Physical Server
VM
GuestOS
Jenkins Instance (Node 1)
Online Banking (Project space)
Credit management (Project space)
Loan management (Project space)
Investment and Wealth Mgmt (Project space)
Tax filing (Project space)
Virtualized Hardware (Hard Disk, RAM, NIC)
Firewall
Load balancer
HTTPS
HTTPS
HTTPS
HTTPS
HTTPS
Online banking team
Credit mgmt team
Loan mgmt team
Investment and wealth mgmt team
Tax filing team

[1] firewall image generated with the help of Microsoft Copilot (SCU organization). Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).
[2] load balancer image generated, cropped to fit the requirement for the slide, using Microsoft Copilot (SCU organization) AI tool. Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).

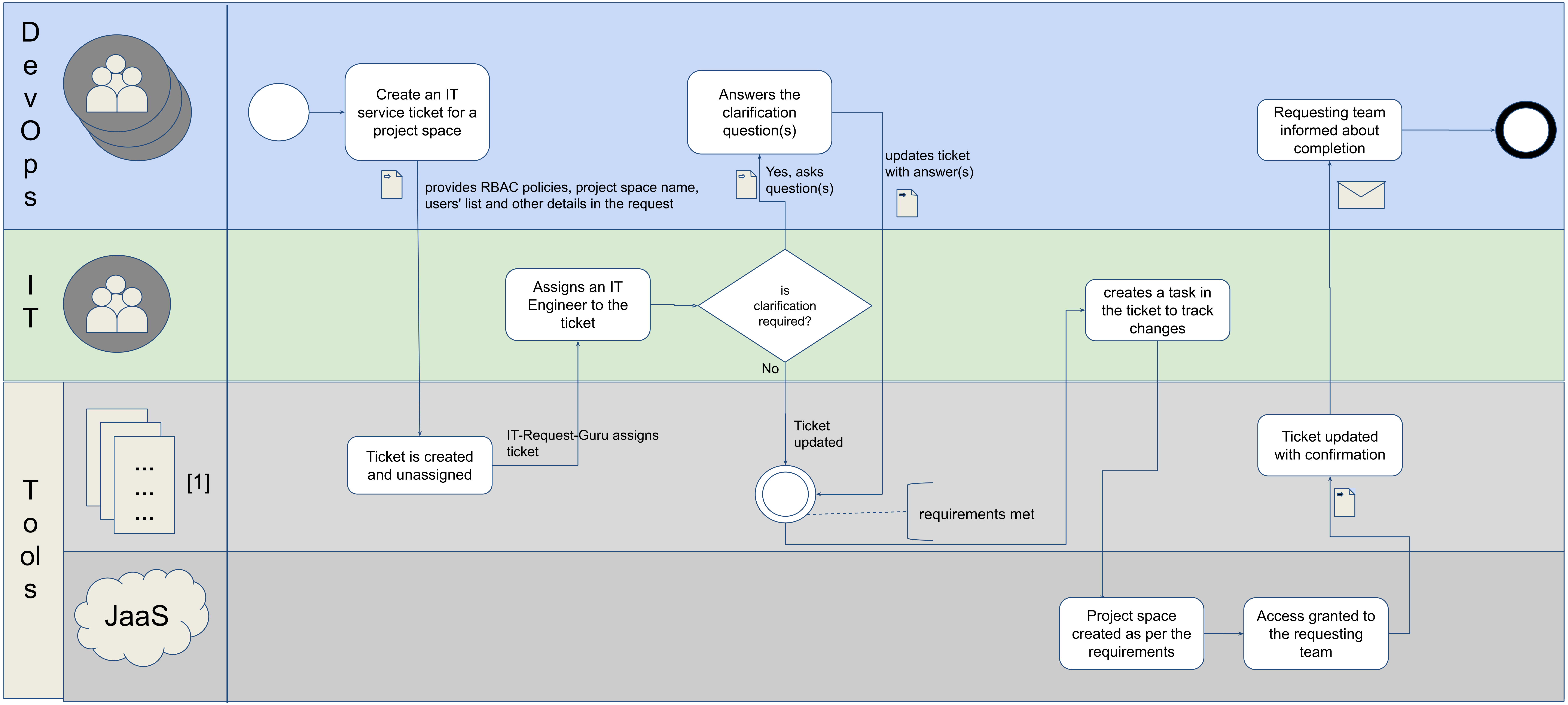
SIMPLE

# TO-BE ARCHITECTURE (ORGANIZATION STRUCTURE)


CEO
CIO / CTO
CISO
COO
10 DevOps are transitioned to:
5 - SRE; 2 - IT; 3 - Security analysts
Open requisitions are closed to offset the transition, ensuring a budget-neutral shift while supporting increased JaaS operational workload.
Director of Engineering
Director of IT
Security Analysts (13)
Marketing (5)
Customer Support(10)
Sales (15)
Operations (4)
IT Engineer (12)
Target Audience
JaaS ownership
Online Banking Engg (6 - DevTest)
Loan App engg (6 - DevTest)
Credit Mgmt Engg (6 - DevTest)
Tax filing Engg. (6 - DevTest)
Investment and wealth Engg. (6 - DevTest)
Site Realiability Engineering (10 - SRE)
JaaS, Git, Selenium
JaaS, Git, Selenium
JaaS, Perforce, Cypress
JaaS, Perforce, Selenium
JaaS, Git, Selenium
JaaS, Git, Perforce, Cypress, Selenium

SIMPLE

# HIGH-LEVEL DESIGN (BPM)


DevOps
IT
Tools
Create an IT service ticket for a project space
provides RBAC policies, project space name, users' list and other details in the request
Answers the clarification question(s)
Yes, asks question(s)
updates ticket with answer(s)
Requesting team informed about completion
Assigns an IT Engineer to the ticket
is clarification required?
No
creates a task in the ticket to track changes
[1]
Ticket is created and unassigned
IT-Request-Guru assigns ticket
Ticket updated
requirements met
Ticket updated with confirmation
JaaS
Project space created as per the requirements
Access granted to the requesting team

[1] Change Management Tracking (IT ticketing system)

SIMPLE

iSync

# LOW LEVEL DESIGN (PROJECT PROVISIONING WORKFLOW)

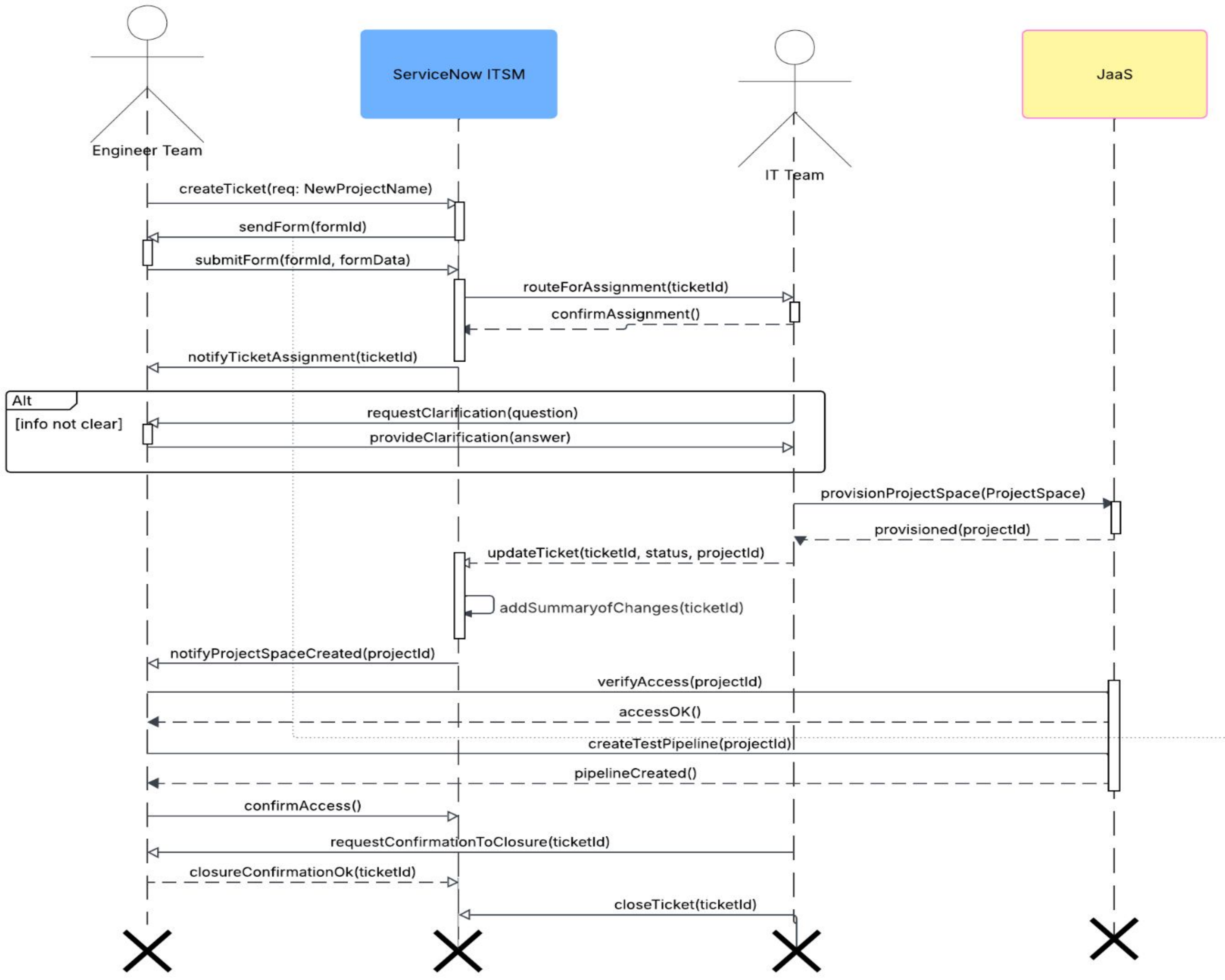
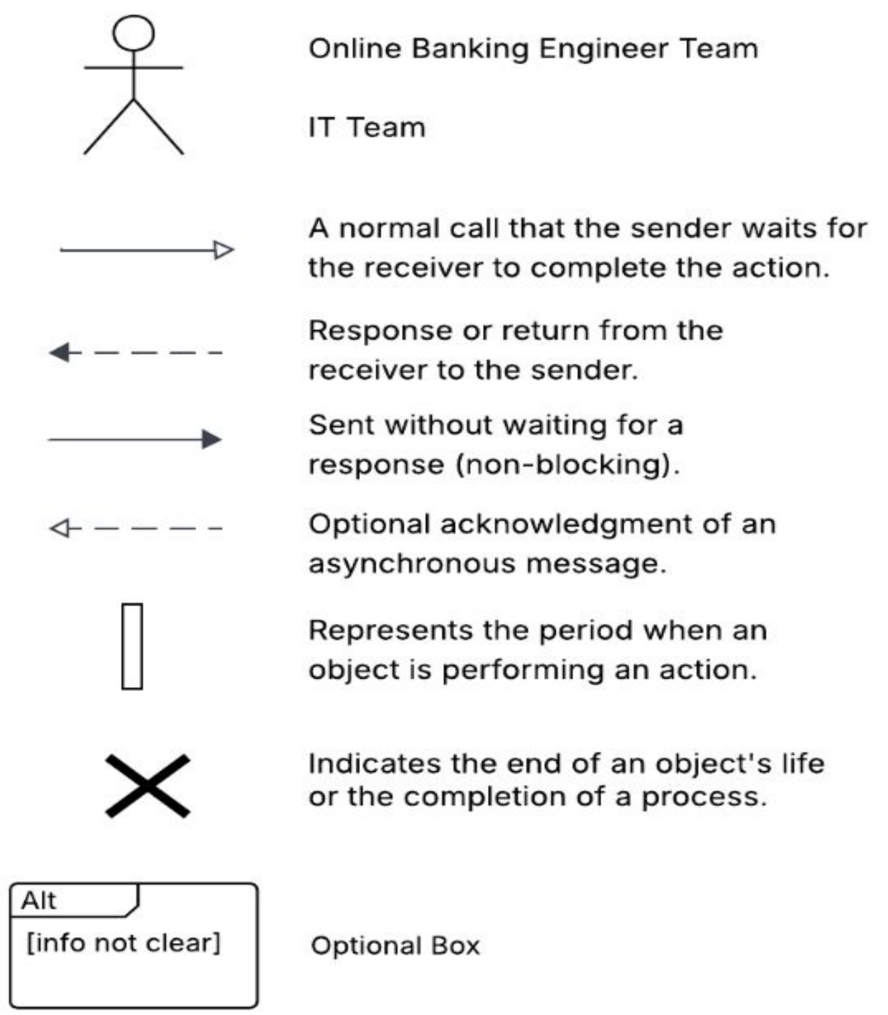

Engineer Team
ServiceNow ITSM
IT Team
JaaS
createTicket(req: NewProjectName)
sendForm(formId)
submitForm(formId, formData)
routeForAssignment(ticketId)
confirmAssignment()
notifyTicketAssignment(ticketId)
Alt
[info not clear]
requestClarification(question)
provideClarification(answer)
provisionProjectSpace(ProjectSpace)
provisioned(projectId)
updateTicket(ticketId, status, projectId)
addSummaryofChanges(ticketId)
notifyProjectSpaceCreated(projectId)
verifyAccess(projectId)
accessOK()
createTestPipeline(projectId)
pipelineCreated()
confirmAccess()
requestConfirmationToClosure(ticketId)
closureConfirmationOk(ticketId)
closeTicket(ticketId)

**Diagram Legend**



Online Banking Engineer Team

IT Team

A normal call that the sender waits for the receiver to complete the action.

Response or return from the receiver to the sender.

Sent without waiting for a response (non-blocking).

Optional acknowledgment of an asynchronous message.

Represents the period when an object is performing an action.

Indicates the end of an object's life or the completion of a process.

Optional Box

**Form Details**

**Section 1 - Application Details:**
1. Requester Name
2. Department/Team
3. Application Name
4. Environment Type (Dev/Test/Prod/UAT)
5. Business Justification
6. Expected Go Live Date

**Section 2 - Acess & Security:**
1. Primary Owner
2. Group Member (RBAC)
3. Access Level
4. RBAC Policy Template

**Section 3 - Technical Requirements:**
1. Build Agents Required
2. Resource Intensity (Low/Med/High)
3. Dependencies
4. Storage Requirements
5. Pipeline Template Requested

**Section 4 - Review & Approvals**
1. DevOps Validation
2. SRE Review
3. Request Approval

SIMPLE

# RACI

| Architecture Task Group | CIO / CTO | Director of IT | IT Engineers (ISync) | CISO / Security Office | Director of Engineering | DevOps Teams |
|---|---|---|---|---|---|---|
| **1. Define JaaS Architecture Blueprint** | **A** | **R** | C | C | C | I |
| **2. Infrastructure Setup & Pilot Migration** | C | **A** | **R** | I | I | C |
| **3. Plugin Security & Governance Policies** | C | R | C | **A** | I | I |
| **4. CI/CD Integration Across Teams** | **A** | C | **R** | C | **C** | **R** |
| **5. Compliance & Audit Automation** | C | I | R | **A** | C | C |
| **6. Enterprise Rollout & Training** | **A** | **R** | **R** | C | C | **R** |
| **7. Monitoring, KPI, and Optimization** | **A** | **R** | R | C | C | C |

R (Responsible) – Executes the task / delivers the output
C (Consulted) – Provides input, review, or validation
A (Accountable) –Owns the final outcome / key decision maker.
I (Informed) – Updated on progress or outcomes.

**CIO / CTO** – Defines and approves the overall JaaS architecture strategy and alignment with enterprise DevOps standards.

**Director of IT** – Accountable for end-to-end implementation and infrastructure resource management.

**IT Engineers (ISync Team)** – Execute core Jenkins deployment, pipeline migration, and integration activities.

**CISO / Security Office** – Accountable for defining and validating compliance, RBAC, and security governance.

**Director of Engineering** – Ensures alignment with SDLC processes and integration with team workflows.

**DevOps Teams** – Consume Jenkins-as-a-Service, participate in rollout and feedback cycles.

ISync

# OSS RESEARCH

SIMPLE

ISync

## OSS OVERVIEW

The CI/CD market[1] has become foundational to modern DevOps[2], enabling continuous integration, testing, and deployment across distributed teams. Organizations are increasingly adopting automation-driven CI/CD pipelines to reduce release cycles, improve reliability, and support cloud-native delivery. Jenkins, GitLab, CircleCI, and GitHub Actions dominate this ecosystem due to their integration depth and scalability.

### *Key market trends include [3]:*

- **Cloud-native adoption:** CI/CD is shifting from on-prem to Kubernetes-native pipelines for elasticity.
- **Security integration:** DevSecOps tools are being embedded into pipelines for SAST/DAST scanning, as ISync plans in its Jenkins-as-a-Service rollout.
- **AI-driven automation:** Predictive build failure detection and automated rollback mechanisms are emerging.
- **Platform engineering:** Enterprises increasingly provide CI/CD as internal platforms (like ISync's Jenkins-as-a-Service) to standardize tooling and compliance

[1] The State of CI/CD Report 2024. Oshyn. Retrieved from https://www.oshyn.com/blog/ci-cd-report-devops
[2] CI/CD in DevOps Workflows. BairesDev. Retrieved from https://www.bairesdev.com/blog/what-is-ci-cd-in-devops/
[3] DevOps vs CI/CD: Key Differences Explained for Modern Software Delivery. Folio3. Retrieved from https://cloud.folio3.com/blog/cicd-vs-devops/

BACKUP

# COMPETITOR ANALYSIS

| Tool | Model | Pros | Cons vs Jenkins | Internal Customer Feedback |
|---|---|---|---|---|
| **GitLab CE** | Integrated DevOps suite (SCM + CI/CD) | All-in-one DevOps workflow; excellent SCM integration; built-in container registry; strong UI/UX; native Kubernetes support. | Tight GitLab-only ecosystem; limited plugin marketplace; customization requires enterprise version; slower feature rollout for CE users. | Teams using GitLab as their main VCS and prefer an integrated DevOps platform over modular CI tools. |
| **GitHub Actions** | Cloud SaaS CI/CD | Seamless GitHub repo integration; YAML-based pipelines; fast setup; strong marketplace actions; scalable cloud runners. | Vendor lock-in to GitHub; limited self-hosting; less control over build environments; lacks deep plugin extensibility and governance flexibility. | Small-to-medium teams already on GitHub seeking minimal CI/CD setup with minimal infrastructure management. |
| **Circle CI Server** | Self-hosted enterprise CI/CD | Solid, enterprise-grade UI; centralized pipeline orchestration; first-class VCS support (GitHub, Bitbucket); scalable runner model; strong analytics dashboards. | Proprietary license; limited plugin ecosystem; slower innovation; less extensible and customizable than Jenkins; less transparent community governance. | Enterprises needing high availability CI/CD with vendor support and predictable maintenance overhead. |
| **Jenkins(Chosen)** | OSS self - hosted | Completely open-source (MIT license); extensive plugin ecosystem (~2000+); integrates with any SCM, cloud, or deployment tool; auditable; flexible pipeline as code; large community and CDF governance. | Requires internal governance, plugin validation, and maintenance; initial setup complexity; UI/UX less modern; requires DevOps expertise for scaling. | Organizations wanting full control, internal compliance, and custom CI/CD governance (e.g., ISync). |

[1] Michael Belton, Alternatives to Jenkins. Retrieved from: https://buildkite.com/resources/blog/alternatives-to-jenkins/
[2] James Walker, Top 10 Most Popular Jenkins Alternatives for DevOps in 2025. Retrieved from: https://spacelift.io/blog/jenkins-alternatives
[3] Sense, Market Share of Jenkins. Retrieved from: https://6sense.com/tech/continuos-integration/jenkins-market-share
[4] BrowserStack, Difference between Jenkins vs Gitlab CI. Retrieved from: https://www.browserstack.com/guide/jenkins-vs-gitlab
[5] Hiren Dhaduk, Jenkins vs CircleCI, Retrieved from: https://medium.com/@HirenDhaduk1/jenkins-vs-circleci-which-is-the-best-ci-cd-tools-558bbe447ccc

COMPLEX

# JENKINS RESEARCH[1]

## Sponsorship & Governance

- Jenkins is funded and governed by the **CDF / Linux Foundation** with corporate sponsors like **CloudBees, Red Hat, and AWS**.
- ISync mirrors this structure internally: its Platform Team acts as an internal governance board approving plugins, validating LTS releases, and auditing security.
- Future goal: contribute a **"JaaS Operator" plugin** upstream to benefit the wider community.
- Example of OSS Upstream: Jenkins Core (MIT licensed project maintained by the Jenkins community).
- Example of OSS Downstream: Jenkins LTS and Jenkins X extend the core for stable and enterprise use cases.

Sponsorship & Governance
Internal Governance
License Compliance

## Legal Permit / License Compliance

- Jenkins is released under the **MIT License**, a **permissive open-source license** that encourages adoption, modification, and redistribution with minimal restrictions.
- Note: "Permissive" means no copyleft obligations — ISync can modify and deploy internally but cannot incorporate GPL-licensed plugins due to Jenkins' governance and license boundary.

## ISync's Rights and Obligations:

- **Full modification rights:** ISync can customize Jenkins and plugins internally to meet enterprise CI/CD needs.
- **Internal deployment freedom:** No obligation to publish internal code or configurations.
- **Attribution requirement:** When redistributing or publishing externally, ISync must retain the original **MIT license notice** and author credits.
- **No copyleft constraints:** MIT licensing does not require ISync to open-source its proprietary extensions, ensuring **compliance flexibility**.
- **Compatible with other permissive OSS components** (Git, SonarQube, JFrog, Docker) under the same open innovation philosophy.
- **Contrast:** Jenkins (MIT OSS) = Open innovation vs CloudBees CI = proprietary copyright.

## Bottom Line:

Sponsorship keeps Jenkins sustainable, Upstream/Downstream integration drives ISync's innovation, and MIT licensing ensures full legal freedom with compliance to OSS principles.

[1] Jenkins Project. *Governance*. Jenkins.io. Retrieved from https://www.jenkins.io/project/governance/
[2] Jenkins : Working with projects. Jenkins.io. Retrieved from https://www.jenkins.io/doc/book/using/working-with-projects/

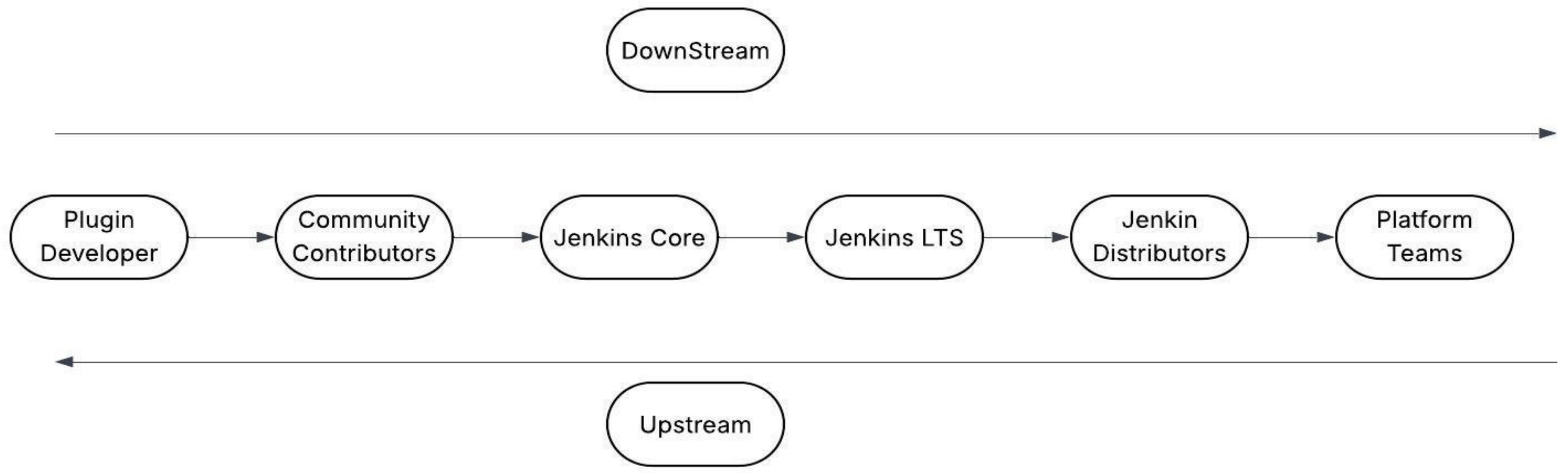
# JENKINS RESEARCH

## Upstream

- Represents the **core foundation** where Jenkins innovation starts - open, collaborative, and community-led.
- Code improvements and bug fixes **flow upward**, strengthening the shared ecosystem.
- Maintained under **MIT's permissive license** and governed by the **Continuous Delivery Foundation (CDF)**.
- Sets the **standards and architecture** that downstream projects build upon.
- Encourages **transparent development** and free contribution from global developers.
- Core takeaway: Upstream drives innovation, quality, and openness across the Jenkins ecosystem.

## MIT-Approved Permissive Downstream

- Built upon the **MIT-licensed Jenkins Core**, allowing enterprises to modify and extend without restriction.
- Enables **controlled distributions** like CloudBees CI and ISync Internal for enterprise environments.
- Acts as a **stabilized layer**, merging upstream updates with internal validation and plugin governance.
- Balances **open innovation and enterprise reliability**, ensuring production-grade quality.
- Supported by CDF governance to maintain **transparency, security, and compatibility**.
- Core takeaway: Downstream turns open-source innovation into secure, scalable enterprise CI/CD solutions.



[1] DevOpsVoyager. *Building Upstream and Downstream Projects in Jenkins: Strengthening Continuous Integration Pipelines.* DevOpsVoyager Hashnode, March 17, 2024. Retrieved from https://devopsvoyager.hashnode.dev/building-upstream-and-downstream-projects-in-jenkins-strengthening-continuous-integration-pipelines
[2] Jenkins : Pipeline: Build Step. Jenkins.io. Retrieved from https://www.jenkins.io/doc/pipeline/steps/pipeline-build-step/

SIMPLE

ISync

## SWOT ANALYSIS[1]

S

### Strengths

1. Highly extensible via plugins and integrations
2. Active global community & long-term support
3. Strong ecosystem of plugins and integrations supported by major enterprises – Jenkins benefits from broad industry adoption, ensuring compatibility and continuous innovation from both community and corporate contributors

W

### Weaknesses

1. Complex plugin management and aging UI
2. Requires manual governance for security
3. Limited scalability and performance under large enterprise workloads – Jenkins can struggle with distributed build coordination and high parallelism without extensive configuration or external tooling [2]

O

### Opportunities

1. Cloud-native modernization through Jenkins X, Kubernetes, and IaC integrations
2. Growing enterprise need for auditable, self-hosted CI/CD due to compliance and data security
3. Potential to contribute innovations (JaaS Operator plugin) back to the community

T

### Threats

1. Competition from GitHub Actions, GitLab CE, and CircleCI offering managed CI/CD
2. Risk of plugin fragmentation or compatibility lag with newer tech stacks
3. Resource overhead for internal governance and maintenance

[1] Wikipedia. *Jenkins (software).* Retrieved from https://en.wikipedia.org/wiki/Jenkins_(software)
[2] Investopedia. *SWOT: What Is It, How It Works, and How to Perform an Analysis.* Retrieved from https://www.investopedia.com/terms/s/swot.asp

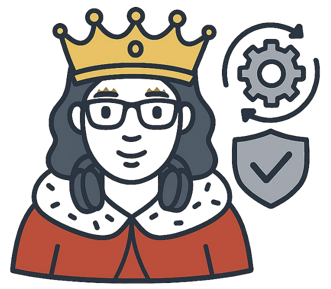
BACKUP

ISync

# PERSONAS & USER STORIES

SIMPLE

# USER PERSONA[1]



[4]

**Name:** Shauna Lee – "The Automation Queen"

**Role:** Senior DevOps Engineer, Online Banking Application Team

**Description:**

Shauna Lee is a senior DevOps Engineer with over 8 years of experience in build and deployment automation for the online banking platform. Her team currently manages their own CircleCI instance customized for project needs but faces repetitive maintenance, patching, and compliance audit overhead. As her application undergoes multiple certifications (SOC 2 [3], ISO 27001, PCI DSS) annually, Shauna collaborates frequently with internal security auditors to gather evidence and ensure adherence to NIST cybersecurity frameworks [2].

With the introduction of Jenkins-as-a-Service (JaaS), Shauna is cautiously optimistic. She hopes to gain better automation, integration, and audit traceability — but worries about slower support cycles and added bureaucracy once Jenkins becomes centralized.

**Functional Requirements (What)**

| Priority (Kano) | Description |
|---|---|
| **Basic** | Enable migration of existing CircleCI pipelines to Jenkins-as-a-Service with minimal disruption |
| **Basic** | Ensure support for SOC 2, ISO 27001, and PCI DSS audit logging in CI/CD jobs |
| **Performance** | Provide reusable pipeline templates with modularized stages for testing, security scans, and deployment |
| **Attractive** | Offer self-service job onboarding with UI-based configuration (no manual YAML editing) |
| **Indifferent** | Allow advanced plugin customization beyond IT-approved catalog |

**Nonfunctional Requirements (How)**

| Priority (Kano) | Description |
|---|---|
| **Basic** | Pipelines must meet corporate security baselines and pass audit validation |
| **Performance** | Job execution time improved by at least 15% compared to CircleCI |
| **Attractive** | Integrated compliance dashboard showing build evidence history per quarter |
| **Indifferent** | Manual upload of audit reports outside the Jenkins environment |

[1] Justin's Team Class Sharing
[2] NIST Cybersecurity Framework (NIST SP 800-53 Rev. 5) . Retrieved from: https://www.forescout.com/ebook-how-to-align-with-the-nist-cybersecurity-framework
[3] SOC 2 Type II & ISO 27001 Standards. Retrieved from: https://info.thoropass.com/soc-2
[4] Persona image generated with the help of Microsoft Copilot (SCU organization). Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).

SIMPLE

# USER PERSONA

**Name:** Joe Park – “The App Guardian”

**Role:** Site Reliability Engineer (SRE)[1], Infrastructure Operations Team

[2]

**Description:**

Joe Park is a Site Reliability Engineer with over 3 years of experience maintaining service reliability and uptime across production environments. Though new to Jenkins, he has deep expertise in incident response and root-cause analysis. Joe has often faced pressure to execute emergency rollbacks and has long advocated for deployment automation to reduce manual errors.

The Jenkins-as-a-Service (JaaS) initiative excites Joe because it promises faster rollbacks and streamlined recovery. However, he worries about the potential complexity and effort required to manage Jenkins servers and the risk of operational delays during critical incidents.

## Functional Requirements (What)

| Priority (Kano) | Description |
|---|---|
| **Basic** | Preserve last 5 stable builds in JFrog Artifactory for reliable rollback |
| **Basic** | Automated rollback pipelines tied to RCA ticketing workflow |
| **Performance** | Parameterized pipelines allowing targeted bug fix deployment |
| **Attractive** | Centralized dashboard visualizing stability metrics and rollback history |
| **Indifferent** | Manual deployment via scripting or SSH access |

## Nonfunctional Requirements (How)

| Priority (Kano) | Description |
|---|---|
| **Basic** | Rollback execution time under 5 minutes for critical services |
| **Performance** | Automated cleanup of builds older than the last 5 versions |
| **Attractive** | Real-time rollback notifications integrated with monitoring tools |
| **Indifferent** | Manual verification of rollback success for low-priority systems |

[1] AWS, What is Site Reliability Engineering (SRE)? Retrieved from: https://aws.amazon.com/what-is/sre/
[2] Persona image generated with the help of Microsoft Copilot (SCU organization). Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).

SIMPLE

ISync

# USER PERSONA

[2]

**Name:** Brian Chen – "The Wizard"

**Role:** IT Engineer, Infrastructure and Platform Services Team

**Description:**

Brian Chen is an IT Engineer dedicating 50% of his time to building and maintaining ISync's Jenkins-as-a-Service (JaaS) platform. He is responsible for provisioning Jenkins project spaces for DevOps teams, enforcing role-based access control (RBAC)[1], and ensuring that every instance complies with organizational security and governance standards.

Brian's mission is to deliver a secure, scalable, and compliant Jenkins infrastructure that replaces fragmented CI/CD setups across teams. He values automation, configuration validation, and continuous compliance monitoring to prevent unauthorized changes and simplify audits.

**Functional Requirements (What)**

| Priority (Kano) | Description |
|---|---|
| **Basic** | Configure centralized RBAC policies to isolate team access and enforce governance |
| **Basic** | Apply approval workflow for plugin installation and configuration changes |
| **Performance** | Validate Jenkins configurations against corporate security baselines |
| **Attractive** | Provide VM provisioning as a service, integrated into Jenkins pipelines |
| **Indifferent** | Allow direct configuration modification by non-IT personnel |

**Nonfunctional Requirements (How)**

| Priority (Kano) | Description |
|---|---|
| **Basic** | Perform automated security scans on Jenkins instances and plugins |
| **Performance** | Maintain audit logs for 90 days and alert on unauthorized changes |
| **Attractive** | Real-time compliance and health dashboard for Jenkins clusters |
| **Indifferent** | Manual configuration reviews outside automated governance framework |

[1] Ferraiolo, D.F. & Kuhn, D.R. (October 1992). "Role-Based Access Control" (PDF). 15th National Computer Security Conference: 554–563
[2] Persona image generated with the help of Microsoft Copilot (SCU organization). Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).

SIMPLE

# USER PERSONA

**Name:** Sam Lee – “The Sentinel”

**Role:** Security Analyst, Cybersecurity & Compliance Team

**Description:**

[1]

Sam Lee is a Security Analyst who has been with ISync since the release of its first online banking platform. He is responsible for evaluating the security of DevOps and SRE tools, defining hardening policies, and managing the company’s regulatory certifications (SOC 2, ISO 27001, PCI DSS). Sam works closely with both DevOps and SRE teams to gather compliance evidence and ensure audit readiness.

As ISync grew rapidly over the past five years, multiple independent DevOps teams adopted diverse CI/CD tools and configurations, making compliance tracking and evidence collection increasingly complex. When Sam learned about the Jenkins-as-a-Service (JaaS) initiative, he was enthusiastic because it would finally standardize CI/CD pipelines, making compliance verification, vulnerability management, and evidence gathering significantly easier and more consistent.

**Functional Requirements (What)**

| Priority (Kano) | Description |
|---|---|
| **Basic** | Enable vulnerability scans for Jenkins plugins to identify risks before installation |
| **Basic** | Implement predefined security baselines automatically applied to all Jenkins instances |
| **Performance** | Centralized compliance evidence collection after each build or configuration change |
| **Attractive** | Provide secure dashboard for real-time access to audit logs and compliance reports |
| **Indifferent** | Manual plugin review processes outside the automated vulnerability scanning framework |

**Nonfunctional Requirements (How)**

| Priority (Kano) | Description |
|---|---|
| **Basic** | Audit logs enabled for all Jenkins configuration and plugin changes, retained for ≥ 90 days |
| **Performance** | Automated compliance validation run monthly against approved security baseline |
| **Attractive** | On-demand compliance audit report generation from stored evidence repository |
| **Indifferent** | Manual evidence submission through email or shared drives |

[1] Persona image generated with the help of Microsoft Copilot (SCU organization). Refer the attached folder for conversation summary and AI tool confirming the image was generated and not sourced from the web).

SIMPLE

# USER STORIES

## Epic 1 – US 01

**As** Shauna, I want to **build a CI/CD pipeline** and not worry about administrative tasks of maintaining the Jenkins instance **so that** I am able to create more capacity to work on providing business value to the online banking application.

- Acceptance criteria:
  - Integration with plugins like git, selenium
  - Replicate the existing pipeline stages in Jenkins-as-a-Service.
  - Manage access control over the pipeline for the online banking team
  - Able to generate capacity by offloading plugin management, CI/CD infrastructure maintenance and administrative tasks of maintaining the CircleCI tool.

**US 01.1**

As a DevOps engineer, I want to replicate my pipelines from CircleCI to Jenkins project space **so that** I can eliminate the burden of maintenance of the CircleCI server and reduce the licensing expense.

- Acceptance criteria:
  - all pipelines are running without any setup error
  - Integrate with git as SCM
  - Integrate with selenium as testing framework and tests are triggered
  - Integrate with a script for VM provisioning where online banking app can be hosted on the Dev-Test environment
  - Pipelines execution and changes can automatically logged for audit reviews

**US 01.2**

As a DevOps engineer, I want to have a project space on jenkins **so that** all the online banking team members can view, modify or run the CI/CD pipeline for our application when we do a code check-in.

- Acceptance criteria:
  - No one outside the Online Banking app team is able to view, run and/or modify the pipelines owned by Shauna's team
  - Access control is managed through JaaS RBAC policies

**US 01.3**

As a DevOps engineer, I want to enable plugins like git, jfrog, selenium on Jenkins instances **so that** I can automatically trigger the test cases upon code check-in.

- Acceptance criteria:
  - DevOps engineers are able to request for a plugin via an IT ticketing system.
  - IT team is able to work with the security analyst to review the plugin health and give it a green signal to be installed
  - IT team is able to deploy the plugin on the jenkins instance and restart jenkins instance

# USER STORIES

## Epic 2 – US 02

**As** Joe, I want to streamline the rollback mechanism **so that** in the event of a disaster Joe is able to provide the least disruption to the customer and adhere to service availability KPI of 99.9% uptime.

- Acceptance criteria:
  - Joe is able to preserve builds
  - Joe is able to deploy a last known stable build based on recommendation from DevOps team.
  - Joe is able to track the rollback via a ticketing system

### US 02.1

As an SRE engineer, I want to preserve the last 5 stable builds using jfrog artefactory **so that** a smooth rollback to a previous stable version in the event of a software disaster can be performed easily without requiring a rebuild of the entire application

- Acceptance criteria:
  - Saving builds are added as a pipeline stage to the production deployment pipeline managed by Joe's team for each application team, in an isolated project space on jfrog artefact repository
  - Builds older than the last 5th deployment are automatically set to delete by using a script
  - No one other than SRE team must have access to the jfrog repository
  - A rollback deployment is always requires a ticket number with root case analysis

### US 02.2

As an SRE engineer, I want to deploy a particular change set approved as a bug fix in the production environment **so that** an important and urgent critical fix can be delivered to the customer faster.

- Acceptance criteria:
  - Pipeline is created to deploy a particular changeset as a parameter, then utilizes this parameter to pick the changes from SCM which need to be deployed to production
  - This pipeline also accepts test suite identifier and build modules to execute targeted build and test modules
  - Every bug fix deployment is associated with a bug number which can be used to trace back the fix

# USER STORIES

**Epic 3 – US 03**

As Brian, I want to ensure that Jenkins instance offered as a service remain secure and compliant with organizational change management policies including implementing security controls, monitoring configurations, and enforcing governance standards **so that** Brian's team can prevent unauthorized changes and maintain adherence to compliance requirements.

- ○ Acceptance criteria:
    - ■ Jenkins instances must have role-based access control (RBAC) configured.
    - ■ All plugin installations and updates require approval through the change management process.
    - ■ Security scans are performed on Jenkins instances and plugins before deployment.
    - ■ Audit logs are enabled and stored for at least 90 days.
    - ■ Automated alerts are triggered for unauthorized changes or failed compliance checks.

**US 03.1**

As an IT engineer, I want to validate Jenkins configurations against security baselines so that compliance is maintained.

- ■ Acceptance criteria:
    - Jenkins instance configuration is checked against the organization’s security baseline document.
    - All required security plugins (e.g., Role-Based Access Control, Audit Trail) are installed and active
    - Unused or deprecated plugins are identified and flagged for removal.
    - Default admin credentials are disabled or replaced with secure credentials.

**US 03.2**

As a DevOps engineer, I want to request plugin installations through a controlled process **so that** only approved plugins are deployed.

- ■ Acceptance criteria:
    - Plugin installation requests must be submitted through the change management system.
    - Requests include plugin name, version, and justification for use.
    - Approval workflow requires sign-off from IT engineer and security analyst.
    - No plugin can be installed without an approved change ticket.
    - Audit trail of request and approval is stored for compliance.

SIMPLE

# USER STORIES

**US 03.3**

As a security analyst, I want to run vulnerability scans on Jenkins plugins **so that** potential risks are identified before installation.

- All requested plugins are scanned using an approved vulnerability scanning tool.
- Scan results include CVE details and severity ratings.
- Plugins with critical or high vulnerabilities are automatically flagged for rejection.
- Scan report is attached to the change request for review.
- Security analyst must approve or reject based on scan results.

**US 03.4**

As a security analyst, I want to review audit logs of Jenkins changes **so that** adherence to change management policies can be verified.

- Acceptance criteria:
    - Jenkins audit logging is enabled for all configuration and plugin changes.
    - Logs include timestamp, user ID, and change details.
    - Logs are stored in a centralized, tamper-proof system for at least 90 days.
    - Compliance officer can access logs through a secure dashboard
    - Monthly compliance reports are generated and archived.

**US 03.5**

As an IT engineer, I want to provide VM provisioning as a service that can be integrated into Jenkins pipelines, so that VMs can be created and destroyed automatically according to company policy.

- Acceptance criteria:
    - Jenkins pipeline can trigger VM provisioning and destruction stages.
    - VM provisioning accepts up to 4 parameters: vCPU, RAM, HDD, NIC.
    - Default values: vCPU=2, RAM=4GB, HDD=8GB, NIC=1.
    - Default OS is Ubuntu.
    - VM is deployed on testing infrastructure and returns a virtual IP.
    - VM lifecycle complies with company policy (auto-destroy after pipeline completion).
    - Parameters are configurable in Jenkins pipeline configuration.

# USER STORIES

**Epic 4 – US 04**

As Sam, I want to ensure that all Jenkins instances provided through JaaS remain continuously compliant with ISync’s cybersecurity and audit requirements by enforcing standardized security baselines and automating compliance evidence collection, so that the Security and Compliance team can maintain certification readiness (e.g., SOC 2, ISO 27001, PCI DSS) and reduce manual effort in gathering and validating audit artifacts across multiple DevOps teams.

- Acceptance criteria:
  - Security baselines are standardized and automatically enforced across all Jenkins instances.
  - Compliance evidence (audit logs, configuration snapshots) is continuously collected and securely stored.
  - Automated compliance validation runs monthly with reporting to the Security team.
  - Only authorized security and IT personnel can modify baseline configurations or access compliance data.
  - Continuous audit readiness dashboards and reports are available on demand.

**US 04.1**

As a security analyst, I want predefined security baselines applied to all Jenkins instances so that hardening policies are consistently enforced across the organization.

- Acceptance criteria:
  - A security baseline document is created and approved, specifying required configurations (e.g., RBAC, HTTPS, audit logging).
  - All Jenkins project space provisioned through JaaS automatically apply the approved security baseline during setup.
  - System validates each Jenkins instance against the baseline on a monthly basis
  - Critical security settings (e.g., admin credentials, encryption) cannot be overridden without an approved change request
  - Compliance reports include proof that baseline configurations were applied and maintained.
  - Only authorized roles can modify or approve baseline configurations.

**US 04.2**

As a security analyst, I want automated collection and storage of compliance evidence (audit logs, configuration snapshots) so that I can easily prepare for security certifications and audits.

- Acceptance criteria:
  - Jenkins instances must have audit logging enabled for all configuration changes and plugin installations.
  - Compliance evidence (audit logs, configuration snapshots) is automatically collected after each pipeline execution or configuration change.
  - Collected evidence is stored in a secure, centralized repository accessible to the security team.
  - Evidence is retained for at least 90 days (or as per company compliance policy).
  - Only authorized roles (security analyst, auditors) can access the evidence repository.
  - System can generate compliance audit reports on demand using collected evidence.

SIMPLE

ISync

# RELATIVE SIZE

| Points | Effort Level | Description | Typical Example |
|---|---|---|---|
| **2 – Very Small** | Minimal | Simple, low-risk configuration or documentation updates. No dependencies or testing required. | Update Jenkins pipeline parameter; minor text change in audit log script. |
| **3 – Small** | Low effort | Small configuration or code change that requires minimal testing. Can be completed within a few hours. | Add plugin to Jenkins instance; configure Git integration for a single project. |
| **5 – Medium** | Moderate | Task involves setup, testing, or limited integration effort. May require collaboration between DevOps and IT teams. | Create a Jenkins project space; migrate a small CI/CD pipeline; basic RBAC setup. |
| **8 – Large** | High | Complex feature or multi-step implementation involving dependencies or multiple systems (VMware, JFrog, etc.). | Implement centralized RBAC policy; enable audit logging across Jenkins instances. |
| **13 – Very Large** | Very High | Highly complex task spanning multiple integrations or requiring significant coordination and validation. Usually broken into sub-tasks or multiple sprints. | Full Jenkins pipeline migration from CircleCI; design and implement VM provisioning service with rollback automation. |

*Note: Scale with Fibonacci sequence (2, 3, 5, 8, 13) [1] [2]*

[1] Why the fibonacci sequence works well for estimating. Retrieved from: https://www.mountaingoatsoftware.com/blog/why-the-fibonacci-sequence-works-well-for-estimating
[2] Dean Leffingwell (2021-07-01), Story, Scaled Agile Framework, retrieved 2022-08-15

SIMPLE

# RELATIVE SIZE

| Factor | Description | Scale (1–5) | Example in JaaS Context |
|---|---|---|---|
| **User / Business Value** | Measures how much benefit the story delivers to users or the organization. | 1 = Low<br>5 = Very High | **Low (1):** Update Jenkins UI text or icon.<br>**High (5):** Centralize CI/CD pipelines to eliminate CircleCI license cost and maintenance effort. |
| **Time Criticality** | Reflects urgency based on deadlines, audit schedules, or dependency impact. | 1 = Not urgent<br>5 = Extremely time-sensitive | **Low (1):** Add optional plugin for non-critical service.<br>**High (5):** Implement RBAC and audit logging required for upcoming SOC2 compliance audit. |
| **Risk Reduction / Opportunity Enablement** | Evaluates how much it reduces risk or enables new capabilities. | 1 = Minimal<br>5 = Major impact | **Low (1):** Create a test report for a single pipeline run.<br>**High (5):** Build automated rollback with JFrog artifacts to prevent downtime and accelerate recovery. |

**WSJF = (Value + Time Criticality + Risk Reduction) ÷ Job Size (Story Points)** [1] [2]

[1] Scaled Agile, Inc. (2023). Weighted Shortest Job First (WSJF). In Scaled Agile Framework (SAFe®) . Retrieved from: https://framework.scaledagile.com/wsjf/
[2] ducalis, WSJF Agile Framework. Retrieved from: https://help.ducalis.io/knowledge-base/wsjf-guide-weighted-shortest-job-first-agile-framework/

SIMPLE

ISync

## RELATIVE SIZE [1]

| Story ID | Persona | Summary | KANO | MOSCOW | Value | Time Criticality | Risk Reduction | Job Size (SP) | WSJF Score | Priority Rank |
|---|---|---|---|---|---|---|---|---|---|---|
| **EPIC 1** | **Shauna** | **Build CI/CD pipeline on JaaS for online banking** | | | **5** | **5** | **4** | **8** | **1.75** | **2** |
| US 01.1 | Shauna | Pipeline migration from CircleCI to Jenkins | Must-be | Must | 5 | 4 | 3 | 8 | 1.5 | 5 |
| US 01.2 | Shauna | Jenkins project space + RBAC for banking team | Must-be | Must | 5 | 5 | 4 | 5 | 2.8 | 3 |
| US 01.3 | Shauna | Enable plugins (Git, JFrog, Selenium) | Performance | Should | 4 | 3 | 3 | 3 | 3.3 | 2 |
| **EPIC 2** | **Joe** | **Streamline rollback and deployment reliability** | | | **4** | **4** | **4** | **8** | **1.5** | **5** |
| US 02.1 | Joe | Preserve last 5 stable builds (JFrog rollback) | Performance | Should | 4 | 4 | 3 | 5 | 2.2 | 4 |
| US 02.2 | Joe | Deploy approved changeset for bug fix | Must-be | Must | 3 | 5 | 4 | 8 | 1.5 | 6 |

[1] ISync team discussion

SIMPLE

ISync

# RELATIVE SIZE

| Story ID | Persona | Summary | KANO | MOSCOW | Value | Time Criticality | Risk Reduction | Job Size (SP) | WSJF Score | Priority Rank |
|---|---|---|---|---|---|---|---|---|---|---|
| **EPIC 3** | **Brian** | **Ensure JaaS security & governance compliance** | | | **5** | **4** | **4** | **8** | **1.65** | **4** |
| US 03.1 | Brian | Validate Jenkins configs vs security baselines | Must-be | Must | 5 | 4 | 4 | 5 | 2.6 | 4 |
| US 03.2 | Shauna | Controlled plugin approval workflow | Performance | Should | 4 | 3 | 4 | 8 | 1.4 | 9 |
| US 03.3 | Sam | Vulnerability scan for Jenkins plugins | Must-be | Must | 4 | 4 | 5 | 5 | 2.6 | 4 |
| US 03.4 | Sam | Review audit logs for compliance verification | Must-be | Must | 5 | 5 | 4 | 5 | 2.8 | 3 |
| US 03.5 | Brian | Provide VM provisioning as a service | Performance | Could | 3 | 2 | 2 | 13 | 0.5 | 10 |
| **EPIC 4** | **Sam** | **Enforce continuous compliance and audit readiness** | | | **5** | **5** | **5** | **5** | **3.0** | **1** |
| US 04.1 | Sam | Enforce predefined security baselines | Must-be | Must | 5 | 4 | 5 | 5 | 2.8 | 3 |
| US 04.2 | Sam | Automate compliance evidence collection | Attractive | Should | 5 | 5 | 5 | 5 | 3.0 | 1 |

SIMPLE

ISync

# PRODUCT BACKLOG

| Epic ID | Epic Name | User Story ID | User Story Summary | Priority | Story Points (SP) | Persona / Owner |
|---|---|---|---|---|---|---|
| **EPIC 1** | CI/CD Pipeline Modernization (JaaS for Online Banking) | **US 01.1** | Migrate pipelines from CircleCI to Jenkins | High | 8 | Shauna |
| | | **US 01.2** | Create Jenkins project space with RBAC | Very High | 5 | Shauna |
| | | **US 01.3** | Enable core plugins (Git, JFrog, Selenium) | Very High | 3 | Shauna |
| **EPIC 2** | Service Reliability & Rollback Automation | **US 02.1** | Preserve last 5 stable builds in JFrog | High | 5 | Joe |
| | | **US 02.2** | Deploy specific changeset for bug fix | High | 8 | Joe |
| **EPIC 3** | Governance, Security & Compliance | **US 03.1** | Validate Jenkins configs vs security baseline | High | 5 | Brian |
| | | **US 03.2** | Controlled plugin approval workflow | Medium | 8 | Shauna |
| | | **US 03.3** | Scan Jenkins plugins for vulnerabilities | High | 5 | Sam |
| | | **US 03.4** | Review Jenkins audit logs | High | 5 | Sam |
| | | **US 03.5** | VM provisioning as a service | Medium | 13 | Brian |
| **EPIC 4** | Continuous Compliance & Audit Automation | **US 04.1** | Enforce predefined security baselines | Very High | 5 | Sam |
| | | **US 04.2** | Automate compliance evidence collection | Very High | 5 | Sam |

BACKUP

# PLANNING

SIMPLE

ISync

# PRODUCT ROADMAP

| Month 1 | Month 2 | Month 3 | Month 4 |
|---|---|---|---|
| Sprint 1-2 (4 weeks) | Sprint 3-4 (4 weeks) | Sprint 5-6 (4 weeks) | Sprint 7-8 (4 weeks) |
| M1 | M2 | M3 | M4 |

| Key Milestone Per Increment | Expected Outcome | How To Measure Success |
|---|---|---|
| M1: Infrastructure Setup & PoC Deployment | 1. Jenkins master & agents provisioned<br>2. PoC pipeline executes successfully with RBAC + basic audit logging | ● Jenkins environment up in Dev<br>● Sample pipeline builds successfully<br>● DevOps users can authenticate and access Jenkins |
| M2: Pilot Migration Phase (CircleCI → Jenkins) | 1. Two pilot pipelines migrated to Jenkins<br>2. Shared Jenkins libraries established | ● 2 pilot pipelines migrated successfully<br>● Build success rate ≥ 95% in Jenkins<br>● No critical or blocker defects reported during pilot |
| M3: Core Plugin Governance & Security Hardening | 1. Plugin approval workflow implemented<br>2. Security baselines applied across Jenkins (RBAC + audit logs) | ● Approved plugin list published<br>● RBAC enforced on all Jenkins nodes<br>● No critical security vulnerabilities |
| M4: Enterprise Rollout & Compliance Validation | 1. Jenkins rolled out to 5 DevOps teams<br>2. Compliance evidence automation functional | ● 80% builds automated via Jenkins<br>● ≥ 5 teams onboarded<br>● Compliance validation completed (baseline + audit logs) |

AVERAGE

ISync

# PRODUCT MILESTONES

Each Sprint = 2 weeks | Hybrid of Scrum and Incremental Sequential

**Sprint1**

US 01.2 – Create Jenkins project space with RBAC configuration.

US 01.3 – Enable core plugins (Git, JFrog, Selenium) for CI/CD setup.

US 03.1 – Validate Jenkins configurations against security baseline.

US 03.4 – Initiate audit logging framework for DevOps activities.

**Sprint2**

US 01.1 – Migrate first CircleCI pipeline to Jenkins (pilot)

US 03.2 – Define controlled plugin approval workflow.

US 03.4 – Standardize audit log schema across nodes.

Milestone M1: Infrastructure Setup & PoC Deployment

**Sprint3**

US 01.1 – Migrate second pilot pipeline and build shared Jenkins libraries.

US 02.1 – Preserve last 5 stable builds in JFrog for rollback validation.

US 03.1 – Validate Jenkins configurations against security baseline.

US 03.4 – Initiate audit logging framework for DevOps activities.

**Sprint4**

US 02.2 – Deploy specific changesets for bug-fix rollouts

US 03.1 – Enforce RBAC policy across all Jenkins nodes.

US 03.5 – Begin VM Provisioning-as-a-Service integration for pipeline environments.

Milestone M2: Pilot Migration Phase (CircleCI → Jenkins)

**Sprint5**

US 02.1 & 02.2 – Integrate rollback automation with monitoring dashboards

US 03.5 – Complete VM provisioning automation (with auto-destroy policy).

US 04.1 – Security Baseline Enforcement (Initiation)

**Sprint6**

US 04.2 – Automate compliance evidence collection and storage.

US 04.1 – Monthly Baseline Validation

Conduct enterprise-wide training on pipeline and governance integration.

Begin rollout across 5 DevOps teams.

Milestone M3: Core Plugin Governance & Security Hardening

**Sprint7**

US 04.1 – Final Integration & Enterprise Regression Validation

Achieve build failure rate < 5% and performance stabilit

**Sprint 8**

US 04.2 – Compliance Evidence Finalization & Dashboard

Conduct final compliance verification & On-demand audit reporting

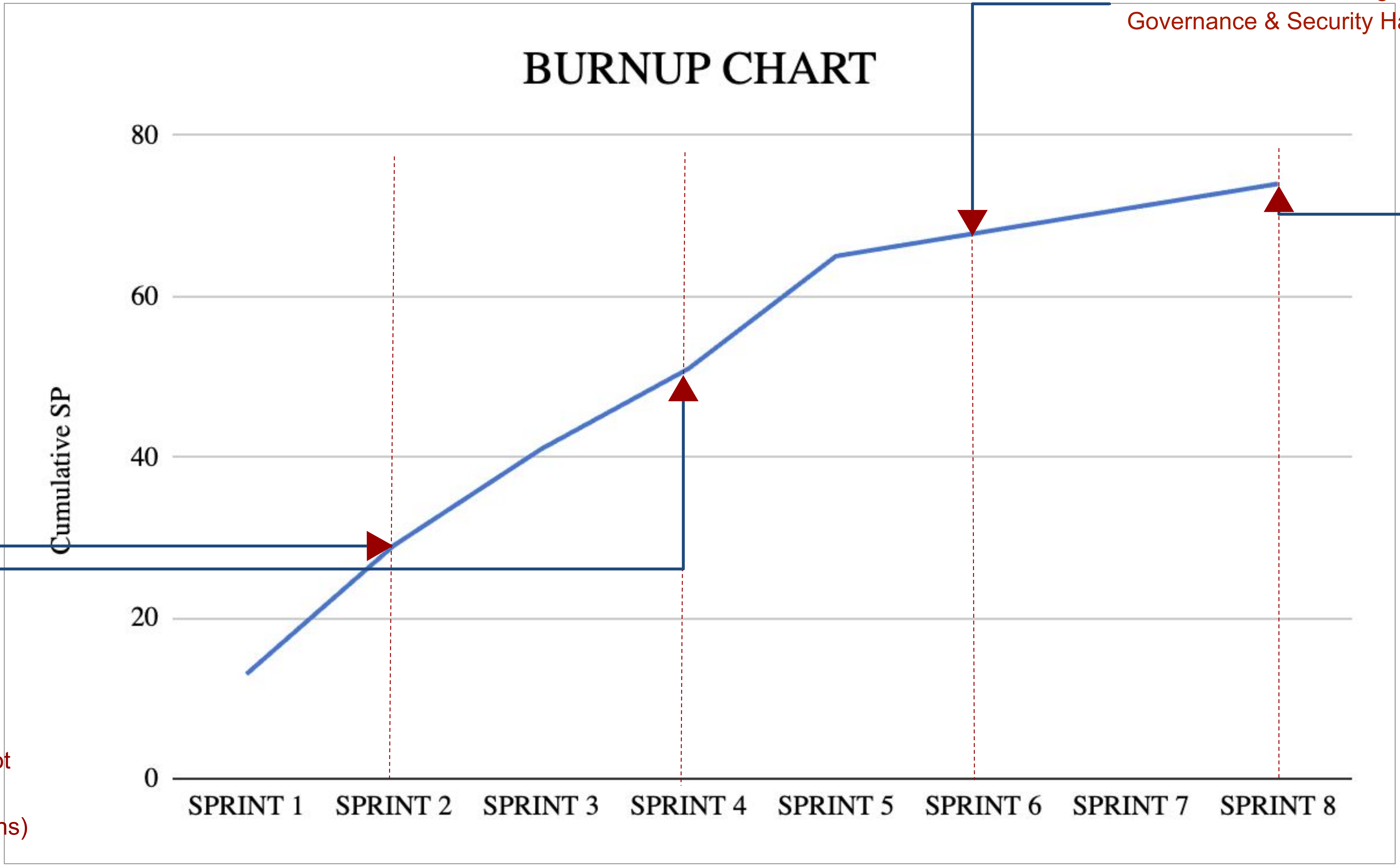
Milestone M4: Enterprise Rollout & Compliance Validation

SIMPLE

# BURNUP CHART

Milestone M3: Core Plugin Governance & Security Hardening

Milestone M1: Infrastructure Setup & PoC Deployment



Milestone M2: Pilot Migration Phase (CircleCI → Jenkins)

After sprint 8, Milestone M4: Enterprise Rollout Phase – > 80% builds automated via Jenkins

SIMPLE

ISync

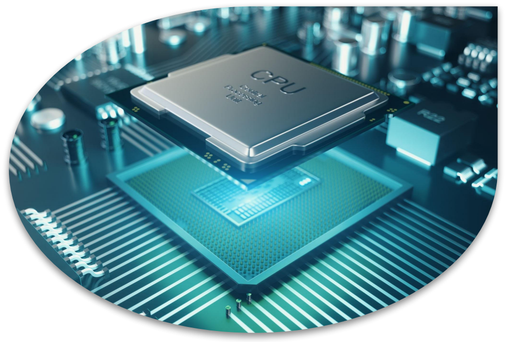
# ANNUAL SAVINGS BREAKDOWN

| | Baseline Assumption | Post JaaS Improvements | Estimated Annual Benefits |
|---|---|---|---|
| **Infrastructure & Maintenance** | **$200,000/yr** 10 VMs for all teams: 10*10000 = $90000 DevOps FTE Cost: 2*0.5 FTE ~ $110000 for plugin drift, config fixes, and disparate monitoring. | **$75,000/yr** Replacing 9VMs with 1 centralized Jenkins Controller + shared agents ~ $20000/yr Lowers DevOps load to 0.5 FTE *(shared admin team) ~$55000* | **$125,000/yr** |
| **License / Plugin Costs** | **$64,000/yr** Multiple CI/CD tools (CircleCI, GitLab, GitHub) with scattered licenses and underutilized seats increase yearly costs | **$33,000/yr** Centralized Jenkins on VMware removes duplicate tools, reduces plugin/admin overhead, and standardizes CI/CD operations. | **$31,000/yr** |
| **Failure Cost Avoidance** | **$108,000/yr** Average ~3 major deployments with ~3 rollback & failure incidents with estimated business cost ~1500/hour 3 incidents × 6 hours × $1,500 × 4 quarters = **$108,000/year** | **$65,000/yr** JaaS standardization + rollback automation reduces failures by 40%. | **$43,000/yr** |

SIMPLE

# ANNUAL SAVINGS BREAKDOWN

**Baseline Assumption** | **Post JAAS Improvements** | **Estimated Annual Benefits**



**CapEx / Hardware Reuse**

**$120,000/yr**

6 CI/CD Servers, 3 Storage Servers, Hardware Refresh Cycle

Mix of depreciation, refresh, and support for all hardware across all teams

**$60,000/yr**

Consolidated deploy servers, unified backup storage, and repurposed 2 servers

**$60,000/yr**

**Engineering Efficiency**

**$1,600,000/yr**

DevOps FTE all inclusive cost

There are 10 intotal DevOps engineer

**$800,000/yr**

Introduction of JaaS, will eliminate the need for 50% of the DevOps engineers.

**$800,000/yr**

| Category | Infrastructure & Maintenance | License / Plugin Optimization | Failure Cost Avoidance | CapEx / Hardware Reuse | Engineering Efficiency | Total Estimated Annual Benefit |
|---|---|---|---|---|---|---|
| **Annual Savings** | $125,000 | $31,000 | $43,000 | $60,000 | $800,000 | **$1,059,000 / year** |

SIMPLE

# COST BREAKDOWN

## INITIAL BUBBLE INVESTMENT (CapEx)

### 1. Jenkins & JaaS Training $-40,000/yr

- Training participants: ~25 engineers (10 DevOps/SRE, 8–10 DevTest, 3–5 IT & Governance)
- Duration: 2 weeks (covering Jenkins-as-a-Service enablement & governance integration)
- External instructor cost: $2,500/day × 5 days = $12,500
- Internal FTE learning time: 25 participants × 2 weeks × $800/week ≈ $20,000
- Lab & pilot environment setup: ~$7,500

### 2. Consultant Support $-20,000/yr

- Certified Jenkins consultant rate: $120–$180/hr (Glassdoor / ZipRecruiter 2024 range)
- Engagement duration: 2 consultants × 2 weeks (≈160 total hours)
- Cost calculation: $125/hr × 160 hrs = $20,000

### 3. Migration & SOP Creation $-10,000/yr

- Effort: 2 DevOps engineers × 1 week
- Internal cost: $2,500/week × 2 = $5,000
- Additional validation & documentation: regression testing, SOP creation ≈ $5,000

## ANNUAL Opex COST

### 1. Infrastructure & Platform Costs $-200,000/yr

- Cloud resources: Jenkins controllers, shared agents, and VM hosting for Dev, QA, and CI/CD pipelines
- Plugin subscriptions: JFrog, SonarQube, GitHub Actions, and monitoring integrations
- Storage & network: Centralized backups, data retention, and bandwidth usage

### 2. Maintenance & Support Costs $-80,000/yr

- Shared DevOps / SRE team: ~0.5 FTE for ongoing patching, upgrades, and configuration maintenance.
- Security and compliance updates: Scheduled audits, access management, and RBAC enforcement.
- Support contracts: Jenkins Enterprise or cloud SLA coverage.

### 3. Monitoring & Tool Management $-40,000/yr

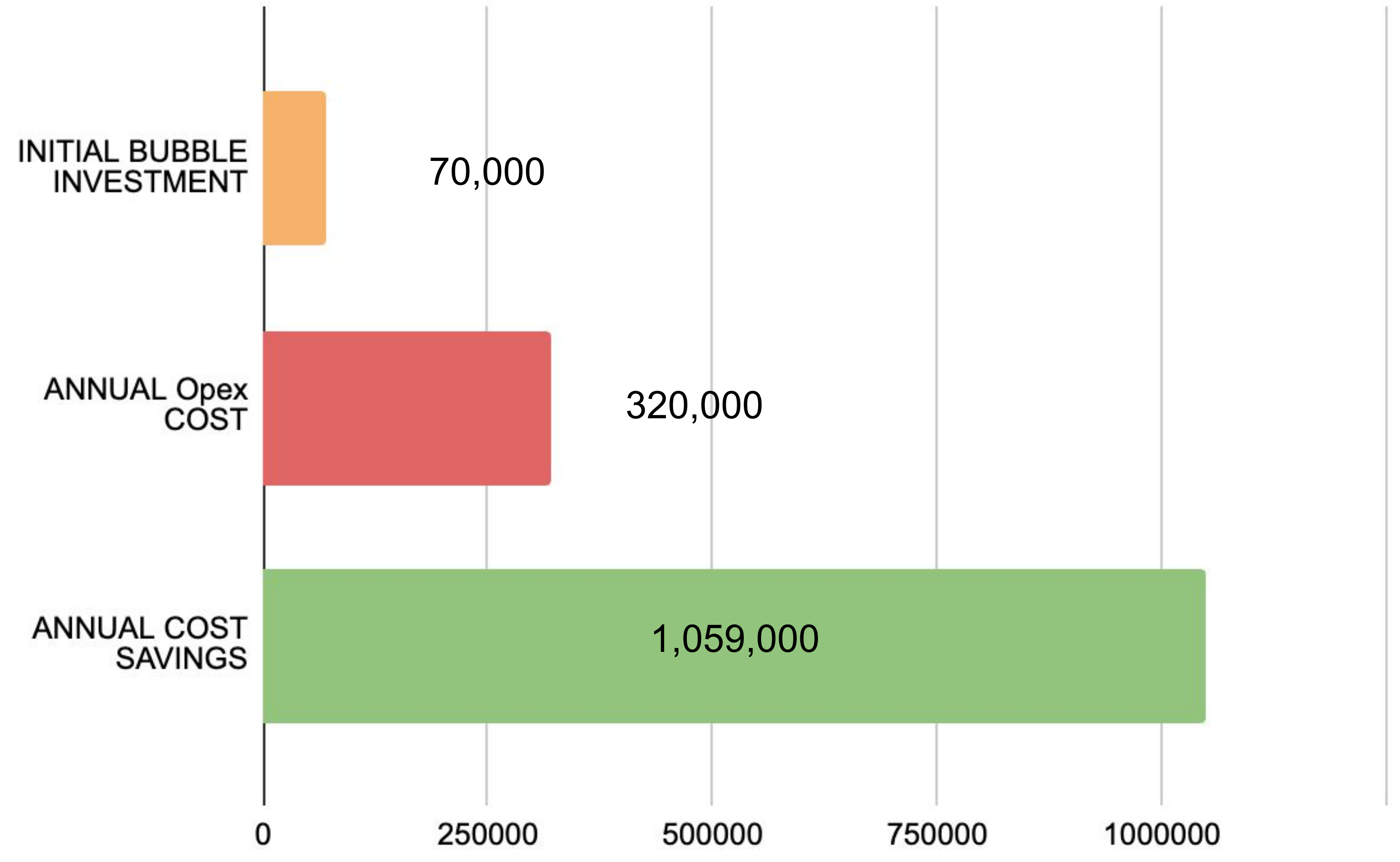
- Monitoring platforms: Grafana / Prometheus / Splunk dashboards for build health and reliability metrics.
- Log management & reporting: Continuous audit trail collection and incident alerting setup.
- Minor administrative overhead: Tool configuration, alerts review, and reporting automation.

SIMPLE

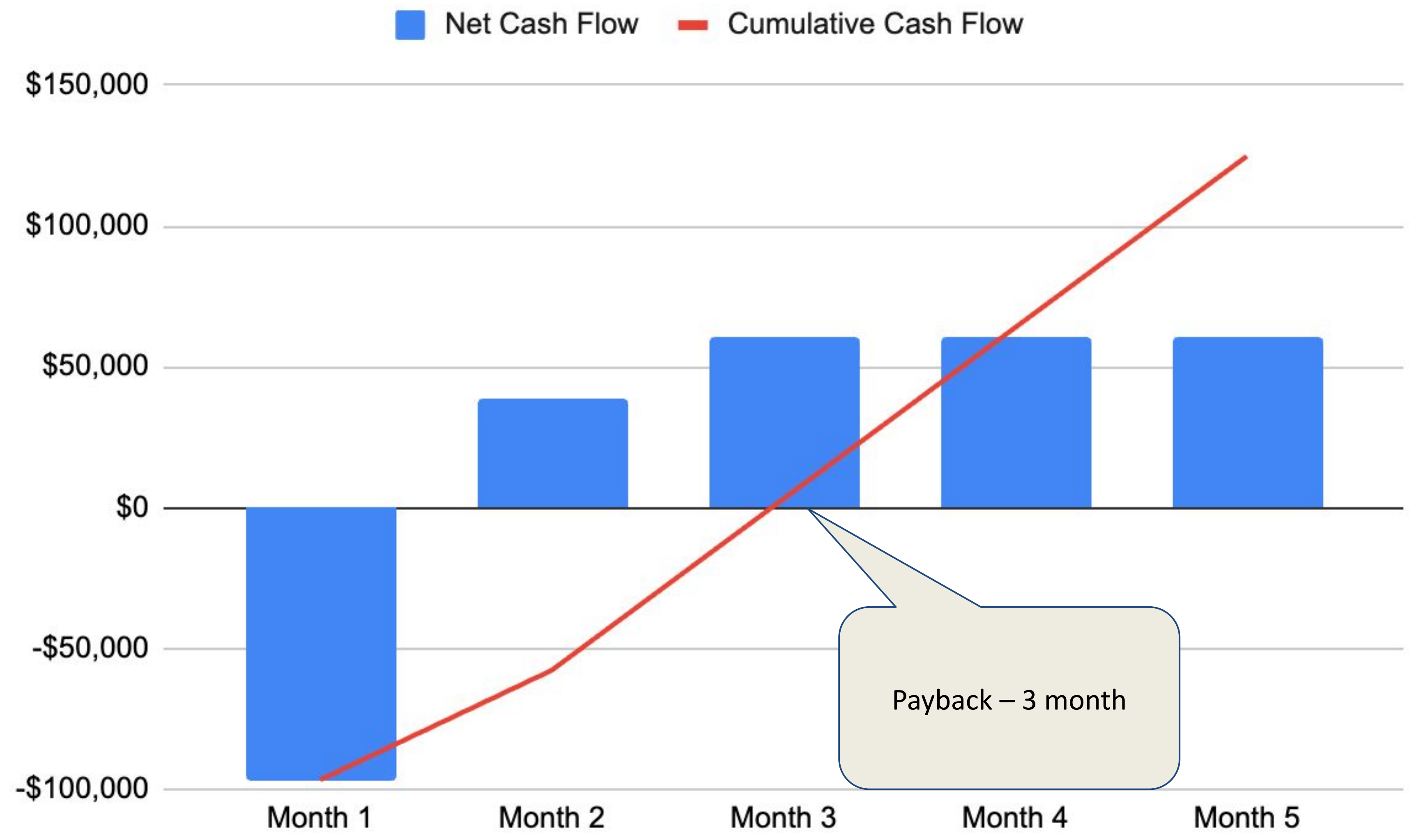
ISync

## ROI & PAYBACK SUMMARY

### Annual perspective



### Monthly perspective



ROI w/o Bubble Cost
= $1059000 - $320000 = $739,000

Payback = ~ 3 month
ROI = ~ 400%

SIMPLE

ISync

# ROV SUMMARY

| Value Dimension | Rationale / Evidence | Expected Outcome |
|---|---|---|
| Developer Productivity & Focus | Developers currently lose 3–4 hrs/week maintaining builds, jobs, diagnosing pipeline drift. With JaaS, this falls to <1 hr/week. (Even if not freeing a full headcount, this time is reclaimed for feature work.) | Faster feature delivery, higher morale, reduced context-switching. |
| Compliance, Audit & Governance | Centralized access control, audit logging, role-based permissions, and consistent pipeline policies make evidence collection easier. This is particularly important for fintech / regulatory domains. | Lower risk of compliance gaps, faster audits, stronger governance posture. |
| Quality / Risk Reduction | Embedding static analysis (SAST), dynamic scans (DAST), dependency checking inside CI can catch vulnerabilities earlier. Industry studies show early defect detection costs far less than late-stage fixes. | Reduced post-release bugs, lower remediation costs, fewer production incidents. |
| Release Stability & Predictability | Standardized pipelines reduce variation and failures across teams. Rollback automation further lowers risk. | More reliable releases, fewer emergency fixes, higher trust in CI/CD. |
| Scalability & Future-readiness | The centralized model is more scalable and easier to maintain as the company grows; avoids explosion of per-team instances. | Better support for new teams, microservices, acquisitions, and growth. |
| Employee Experience & Onboarding Efficiency | With standardized templates and self-service onboarding, new engineers ramp faster without manual scripting. | Reduced onboarding time, lower training overhead. |

# ROV SUMMARY

If workload doubles (2 times) but cost increases only by 1.4× (α = 0.7), the system achieves sublinear cost scaling.

Prior to JaaS, FinTech’s CI/CD costs grew *linearly* with new products and teams. After JaaS, shared pipelines and centralized Jenkins management enable sublinear scaling, meaning the marginal cost of onboarding a new team or service drops over time.

| Year | Teams | Workload | Cost Growth | Efficiency Gain |
|---|---|---|---|---|
| 2025 | 10 | 1000 | Baseline $100K | |
| 2026 | 15 | 1500 (+50%) | $130K (+30%) | +20% |
| 2027 | 20 | 2000 (+33%) | $150K (+15%) | +18% |
| 2028 | 25 | 2500 (+25%) | $165K (+10%) | +15% |
| 2029 | 30 | 3000 (+20%) | $180K (+9%) | +11% |

BACKUP

ISync

# RISK MANAGEMENT

SIMPLE

ISync

## RISK & MITIGATION

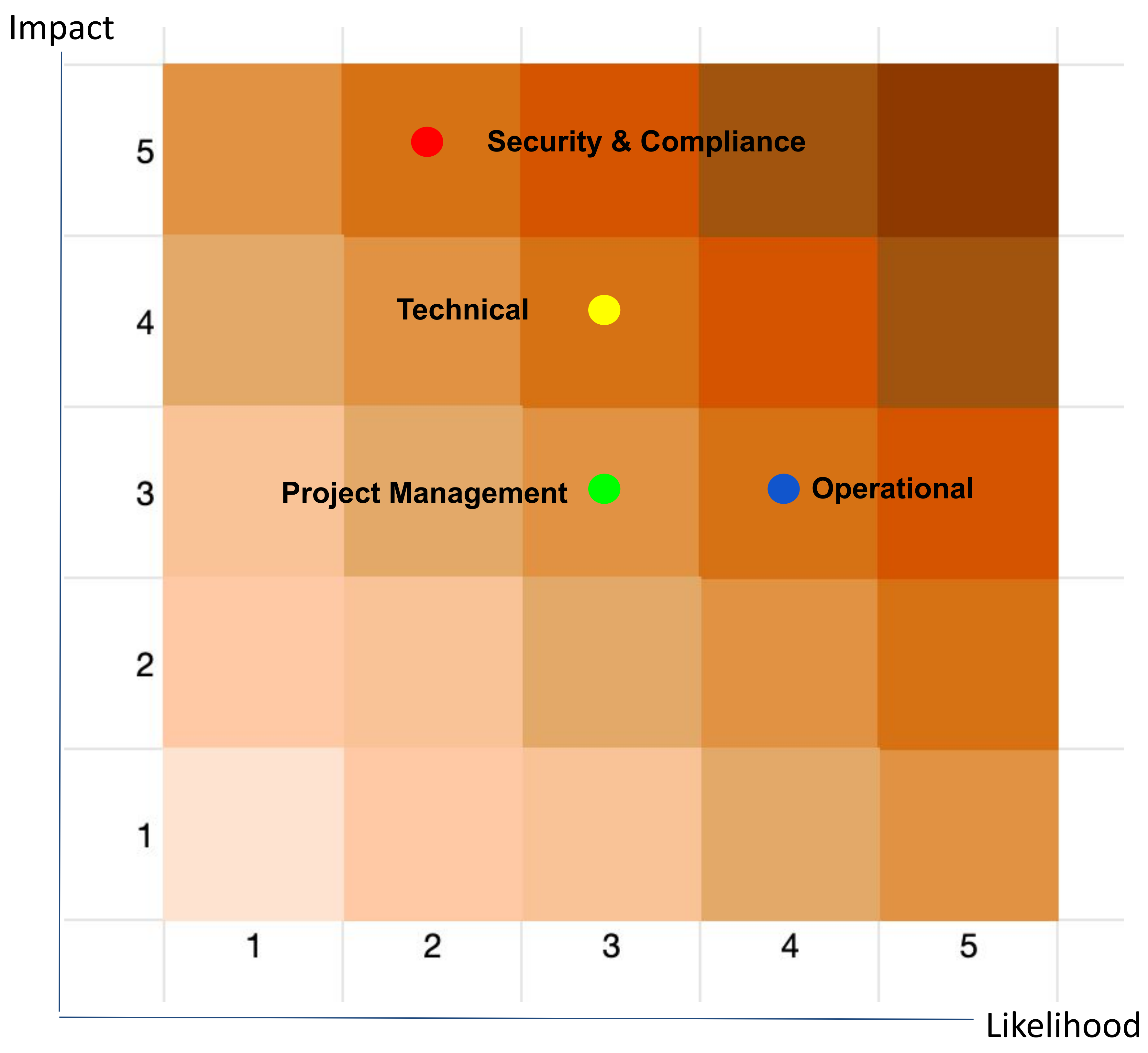
| Category | Description of Risk | Potential Impact | Likelihood Level<br>Very low ‘1’ to Very high ‘5’ | Impact Level<br>Very low ‘1’ to very high ‘5’ | Priority Level (Likelihood x Impact) | Mitigation Strategy |
|---|---|---|---|---|---|---|
| **Technical Risks[1]** | Jenkins or pipeline configuration failures due to misconfigured plugins or updates. | CI/CD processes interrupted; delayed releases and broken builds. | 3 | 4 | 12 | **US01.1 - US01.3**: Standardize CI/CD pipelines in Jenkins-as-a-Service to remove configuration inconsistencies. Automated setup and plugin control ensure consistent with zero errors deployment across teams. |
| **Operational Risks[2]** | VMware or infrastructure downtime causing Jenkins unavailability. | Deployment interruptions; loss of productivity; potential SLA violations. | 4 | 3 | 12 | **US02.1 – US02.2:** Establish automated rollback pipelines and store the last five stable builds via JFrog. This enables rapid recovery and maintains 99.9% service uptime during failures. |
| **Security & Compliance Risks[3]** | Weak RBAC policies or missing audit logs allowing unauthorized access or untracked deployments. | Security breaches, compliance violations, and audit failure during reviews. | 2 | 5 | 10 | **US03.1 – US03.2:** Enforce RBAC baselines and a controlled plugin approval process via ServiceNow ITSM, with automated compliance checks, audit logging, and alerts to prevent unauthorized changes and configuration drift. |
| **Project Management Risks** | Miscommunication between IT and DevOps teams during Jenkins setup or testing. | Schedule delays; unclear ownership; reduced efficiency in coordination. | 3 | 3 | 9 | Hold regular cross-team check-ins; maintain shared documentation; track scope and progress updates through Jira or similar tools. |

[1] Cycode published an article about Jenkins configuration and security risks. Retrieved from: https://cycode.com/blog/jenkins-security-best-practices/
[2] VMware shared best practices for maintaining high availability and minimizing downtime. Retrieved from: https://knowledge.broadcom.com/external/article/313917/best-practices-and-advanced-features-for.html
[3] OWASP Foundation provided guidelines on DevSecOps and RBAC-based security practices. Retrieved from: https://owasp.org/www-project-devsecops-guideline/

SIMPLE

iSync

# RISK HEAT MAP



🔴 **Security & Compliance:** Highest impact risk; requires strong RBAC, audit logging, and periodic reviews.

Highest impact risk, requires strong RBAC, audit logging, and periodic reviews.

This is the risk scenario expanded in the SW Disaster

🟡 **Technical:** Medium risk; prevent Jenkins or plugin failures via config standards and rollback readiness.

🔵 **Operational:** Likely risk from VM or network downtime; mitigate through monitoring and auto recovery.

🟢 **Project Management:** Moderate; improve communication and sprint tracking to reduce delays.

**Overall:** Security & Compliance is the top concern; others are manageable through proactive monitoring and process governance.

SIMPLE

# SW DISASTER

*(Corresponds to: Security & Compliance Risks — Risk #3)*

## Scenario

ISync's centralized Jenkins-as-a-Service faces security and compliance risks when **weak RBAC settings**, **missing audit logs**, or **unauthorized plugin installations** bypass the required approval workflow.
These misconfigurations may result in untracked deployments, failed builds, inconsistent deployment artifacts, audit violations, or potential security exposure[1].

## Impact

Unauthorized plugin installation and configuration drift can cause production pipeline outages, failed deployments, and prolonged service interruptions.
Such failures weaken compliance reliability and may lead to audit findings, SLA violations, and potential data exposure.
Repeated disruptions reduce engineering productivity, damage organizational trust, and may result in significant operational, reputational, and financial impact[2].

## Mitigation Plan

**US03.1 – US03.2:**
We will enforce a controlled plugin approval workflow through ServiceNow ITSM, ensuring all plugin installations and updates receive IT and security approval before deployment. Automated compliance checks will validate Jenkins configurations against required security baselines, detect deprecated or insecure plugins, and ensure mandatory security controls remain active. Audit logs will be retained and monitored, and alerts will notify the team of unauthorized changes, preventing configuration drift and maintaining secure, compliant Jenkins instances across the organization.

[1] Jenkins Role-Based Authorization Strategy Plugin. *Jenkins.io*. Retrieved from: https://plugins.jenkins.io/role-strategy/
[2] Audit Log Best Practices for Security and Compliance. *Digital Guardian*. Retrieved from: https://www.digitalguardian.com/blog/audit-log-best-practices-security-compliance

# RELEASE PLAN

ISync

# VERSION POLICY

1. **Version format: release.major.minor.patch[.env]** [1]
   **Release**:
   - Indicates the platform evolution phase.
   - Increased only when we introduce major capability changes or architectural shifts (new orchestration model, new Jenkins footprint)

   **Major (x)**:Introduced when significant CI/CD capability upgrades or architecture improvements occur [2]
   **Minor (y)**: Backwards-compatible enhancements, including:
   - New features, plugin integrations, workflow improvements
   - Maintenance or "update" releases after pilot rollout (e.g., stability improvements, usability fixes)

   **Patch (z)**:
   - Regular bug fixes and configuration tuning (non-breaking)
   - Different from HotPatch (used for emergency fix)

   **HotPatch (P)**: Emergency fix for production-critical or security incidents
   **Env(optional)**: Environment qualifier for deployments: dev, test, stage, prod

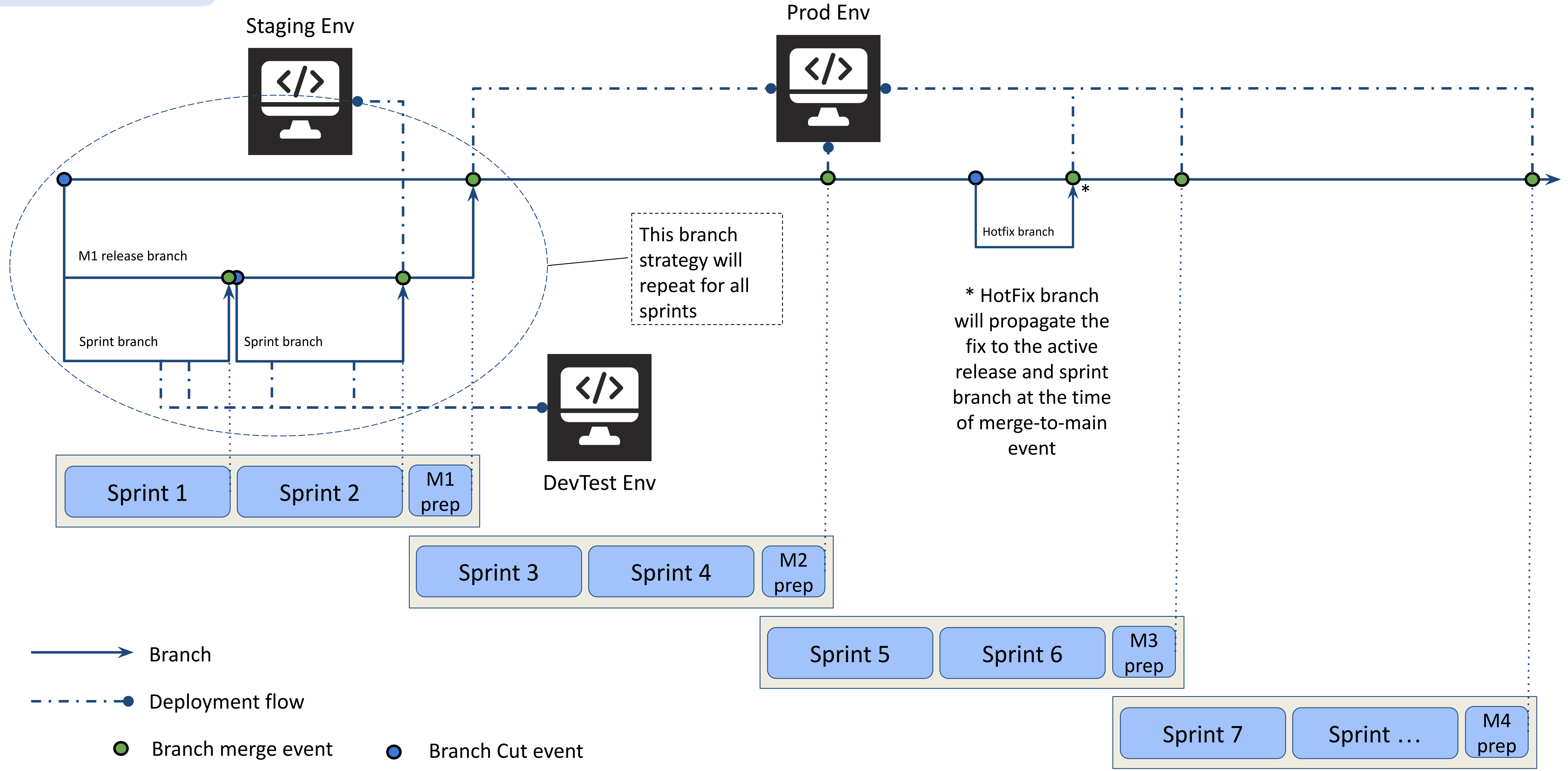
2. **Branching Model:** We follow a multi-stream branching strategy to support parallel feature development, scheduled releases, and unplanned fixes.

   - main → Stable production-ready branch (RBAC-locked; no direct commits). All releases are merged here after validation.
   - release/{version} → Stabilization branch for each planned release (release/v1.0.0, release/v1.1.0, release/v1.2.0, release/v1.3.0).
   - patch/{issue} → Regular maintenance fixes after pilot rollout (version x.y.z updates).
   - hotfix/{issue} → Emergency fix branch for production-critical issues (version x.y.z.P). Hotfix merges propagate to main, active release, and current sprint branches.

[1] AWS, Using Semantic Versioning to Simplify Release Management, https://aws.amazon.com/blogs/devops/using-semantic-versioning-to-simplify-release-management/
{2] Gleeus, The Importance of Major, Minor, and Patch Releases in The Iterative SDLC, https://gleecus.com/blogs/major-minor-and-patch-releases/
[3] Vincent Driessen, A successful Git branching model, https://nvie.com/posts/a-successful-git-branching-model/

SIMPLE

# SDLC, BRANCH & DEPLOYMENT MODEL



SIMPLE

ISync

## RELEASE CRITERIA

| Criteria | Release (v1.0.0) | Release (v1.1.0) | Release (v1.2.0) | Release (v1.3.0) |
|---|---|---|---|---|
| **Functional Readiness & Acceptance** *All prioritized features for this release are validated and meet acceptance criteria.* | 75% Functional requirements passed | 80% Functional requirements passed | 90% Functional requirements passed | 95% Functional requirements passed |
| **Quality & Reliability Stability** *System maintains acceptable defect levels and regression quality.* | ≤ 3 critical defects; Regression suite ≥ 85% pass | ≤ 2 critical defects; Regression suite ≥ 90% pass | ≤ 1 critical defect; Regression suite ≥ 95% pass | Zero critical defects; Regression suite 98% pass |
| **Performance & Scalability** *CI/CD performance meets baseline expectations.* | Build success rate ≥ 90% | Build success rate ≥ 93% | Build success rate ≥ 95% | Build success rate ≥ 98% |
| **Security Compliance & Governance** *All security controls enforced and no critical vulnerabilities remain.* | ≤ 2 P1 security bugs; No P0; OSS CVSS < 7.5 | ≤ 1 P1; No P0; CVSS < 6.5 | 0 security bugs; CVSS < 5.5 | 0 security bugs; CVSS < 5.0 |
| **Operational Readiness & Deployment Stability** *Release is deployable, validated, and ready for production without major risks.* | Sanity test 90% pass; Rollback validated manually | Sanity test 93% pass; Rollback partially automated | Sanity 95% pass; Fully automated rollback | Sanity 98% pass; Rollback auto + audited |
| **Documentation, Legal & Compliance** *Release includes accurate technical docs, user docs, and legal compliance.* | 80% docs completed | 90% docs completed | 95% docs completed | 100% docs completed |

SIMPLE

ISync

# RELEASE CHECKLIST

| Category | Checklist | Person Responsible | Release 1.0.0 | Release 1.1.0 | Release 1.2.0 | Release 1.3.0 |
|---|---|---|---|---|---|---|
| **Requirement Completion** | Confirm all planned epics/stories for this release are completed and reviewed | PM | Y | Y | Y | Y |
| | Validate acceptance criteria coverage | Tester | Y | Y | Y | Y |
| | All P0/P1 requirements must pass functional validation | Tester | | Y | Y | Y |
| **Functional Readiness** | All pipelines can run successfully end-to-end | Tester | Y | Y | Y | Y |
| | Plugin workflow validated (build → test → artifact) | Tester | Y | Y | Y | Y |
| | Shared library and base pipeline templates tested | SRE | | Y | Y | Y |
| **Security & Access Control** | RBAC roles validated for all teams | Security | | Y | Y | Y |
| | Permission boundaries tested according to policy | Security | | Y | Y | Y |
| | Jenkins audit log verified for critical events | Security / DevOps | | | Y | Y |
| **Compatibility & Integration** | Jenkins core version validated with all approved plugins | Security | | Y | Y | Y |
| | Team pipelines validated for compatibility | Tester | Y | Y | Y | Y |
| | Integration with external tools (repo / artifact store / secrets manager) tested | DevOps | | Y | Y | Y |
| **Operations & Deployment Readiness** | Deployment configuration validated in dev/test/stage environments | DevOps | Y | Y | Y | Y |
| | Backup / rollback workflow validated | DevOps | | Y | Y | Y |
| | Monitoring & alerting rules configured | DevOps | | | Y | Y |
| **Documentation & Compliance** | Release notes completed | PM | | Y | Y | Y |
| | User onboarding guide updated | PM | | | Y | Y |
| | All compliance documents reviewed and stored | PM / Security | | | Y | Y |

AVERAGE

ISync

# RELEASE ROADMAP[1]

| | Release (v1.0.0)<br>Month 1 (sprint 1-2) | Release (v1.1.0)<br>Month 2 (sprint 3-4) | Release (v1.2.0)<br>Month 3 (sprint 5-6) | Release (v1.3.0)<br>Month 4 (sprint 7-8) |
|---|---|---|---|---|
| **FOCUS** | **Infrastructure Setup & PoC Deployment** | **Pilot Migration & Shared Library Setup** | **Governance + Security Hardening** | **Rollout Readiness** |
| **COMPONENT** | • Jenkins master & agents provisioned<br>• Core plugins (Git, JFrog, Selenium) enabled<br>• Security baseline validation (initial)<br>• PoC pipeline executed successfully | • 2 pilot teams migrated (CircleCI → Jenkins)<br>• Shared Jenkins libraries built<br>• Audit logging standardized• RBAC enforced across pilot environments | • Plugin approval workflow implemented<br>• Security baselines enforced across JaaS<br>• Automated VM provisioning<br>• Compliance evidence auto-collection (partial)<br>*Includes updates/improvements from M1 & M2 pilot rollout feedback.* | • Final integration & regression validation<br>• Compliance readiness dashboard<br>• Enterprise-wide documentation & training<br>• Compliance evidence automation completed |
| **CHECKLIST** | Requirement Completion ✓<br>Functional Readiness ✓<br>Security Baseline (initial) ✓<br>Ops Readiness (Dev env) ✓ | Compatibility & Integration ✓<br>Audit Logging ✓<br>Shared Libraries ✓<br>*includes all checklist items from v1.0.0* | Security Governance ✓<br>Plugin Governance ✓<br>Compliance Collection (partial) ✓<br>*includes all checklist items from v1.1.0* | Full Compliance Validation ✓<br>Training & Documentation ✓<br>Enterprise Readiness ✓<br>*includes all checklist items from v1.2.0* |
| **TESTING [2]** | Integration Testing<br>Reliability Testing<br>Basic Functional Testing | Functional Testing<br>Security Baseline Validation<br>Performance & Stress Testing | End-to-End Testing<br>Rollback Workflow Testing<br>Security Compliance Spot-Check | Full System Testing<br>User Acceptance Testing (UAT) |

[1] Lecture Slides: ISBA2408 Fall 2025 RelBranchProjMgmt, Release Roadmap, Page 31-33
[2] LARION, Fundamentals of Software Testing: Types, Methodologies, STLC, Models, https://larion.com/software-testing-fundamentals/

ISync

## TESTING PLAN

SIMPLE

# TESTING STRATEGY

## Testing Goal

**Functional Assurance:** JaaS components work correctly end-to-end

**Reliability & Performance:** Pipelines run predictably, rollback is consistent

**Security & Compliance Validation:** RBAC, audit logs, plugin governance

**Value Realization:** Testing confirms productivity and cost benefits

## Testing Scope

- Integrates Git, Selenium, VM provisioning, and CI/CD flows
- Supports pipeline migration from CircleCI
- Enforces centralized RBAC and plugin approval workflows
- Enables rollback, build retention, and change-set deployments
- Meets security, audit, reliability, and infrastructure efficiency targets

## Testing Case

### 1. Boundary Value Analysis (BVA)

- Number of concurrent pipelines (e.g., 1, 50, 100).
- Build retention limits (e.g., 4, 5, 6 builds).
- User roles (no permissions → partial → full permissions).

### 2. Equivalence Partitioning

- Valid vs invalid SCM URLs
- Approved vs unapproved plugins
- Valid vs invalid VM provisioning parameters

### 3. State Transition Testing

Used for RBAC, plugin approval workflow, rollback lifecycle:

- Requested → Under Review → Approved → Installed → Active
- Build Passed → Build Failed → Rollback Triggered → System Stable

### 4. End-to-End Scenario Testing

- Git push → Jenkins build → Selenium tests → VM deploy → Audit logs generated
- CircleCI → Jenkins migration scenario tests

BACKUP

ISync

# TESTING ENVIRONMENT

## 1. Test Data Types :

**Mock Git Repositories**

- Valid repo, invalid repo, inaccessible repo

**Selenium Test Suites**

- Passing tests, failing tests

**VM Provisioning Parameters**

- Valid config, missing config, oversized VM request

**User Accounts**

- SRE, DevOps Engineer, Unauthorized User

**Plugin Requests**

- Approved plugin, blocked plugin, deprecated plugin

## 2. Data Creation Approaches :

**Synthetic Data**

- Sample commits, dummy test files, fake changesets

**Production-Sanitized Data**

- Anonymized real pipelines for realistic behavior

**Event Simulation**

- Simulated build failures, rollback triggers

**Mock APIs / Stubs**

- For unavailable services (e.g., JFrog outages)

## 3. Environment :

- Dedicated Jenkins dev instance
- Isolated project spaces for each persona
- Infrastructure-as-Code templates for consistent setups
- Test datasets loaded automatically via CI/CD

[1] TechTarget. (n.d.). *Guide to Synthetic Test Data*. Retrieved from https://www.techtarget.com/searchsoftwarequality/tip/Guide-to-synthetic-test-data TechTarget
[2] Mason, B. (2025, October). *How to Use Test Data Management Strategies for Effective Testing*. Medium. Retrieved from https://medium.com/@testwithblake/how-to-use-test-data-management-strategies-for-effective-testing-7b23594457d0

SIMPLE

## EPIC 1 — CI/CD PIPELINE

| Type | Acceptance Criterion Being Validated | How It’s Verified (Conceptual) | Success Metric / Outcome |
|---|---|---|---|
| Integration + Smoke Testing | JaaS integrates with Git, Selenium, and existing CI/CD stages;<br>Access control & capacity scaling enabled | Run a full CI/CD flow Git → Jenkins → Selenium and confirm pipeline stages run without manual setup. | ● Pipeline runs without errors<br>● Access control enforced.<br>● No manual setup needed<br>● Capacity scales normally |
| Migration Testing + End-to-End (E2E) Testing | Pipelines migrate from CircleCI → Jenkins; Git + Selenium + VM provisioning script integrated | Deploy known CircleCI pipeline into Jenkins and test SCM linkage, Selenium trigger, VM provisioning, and code-check-in automation. | ● Selenium tests triggered<br>● VM created<br>● Pipeline auto-runs on check-ins<br>● Audit logs captured |
| RBAC Access Control Testing | Only online-banking team can access pipelines;<br>Access managed via JaaS RBAC | Test authorized vs unauthorized access using RBAC role accounts. | ● Unauthorized access denied<br>● RBAC rules applied consistently |
| Security Testing + Approval Workflow Testing + Smoke Testing | Plugin installation follows IT + Security approval workflow;<br>Jenkins restarts cleanly | Simulate plugin request → go through approval → install → validate plugin availability after restart. | ● Only approved plugins installed<br>● Audit trail stored<br>● Jenkins restarts without failure |

SIMPLE

# EPIC 2 — ROLLBACK AND STABILITY

| Type | Acceptance Criterion Being Validated | How It's Verified (Conceptual) | Success Metric / Outcome |
|---|---|---|---|
| Rollback Testing + Stability Testing + Smoke Testing | SRE can preserve builds, deploy last stable build, and track rollbacks. | Simulate failure → trigger rollback pipeline using stored stable build. | ● Rollback succeeds<br>● Build is traceable<br>● Meets 99.9% uptime target |
| Retention Testing + Access Control Testing | Last 5 builds preserved via JFrog; older builds auto-deleted;<br>Only SRE has access. | Upload 7 builds → verify builds 1–2 deleted → test restricted access with non-SRE account. | ● Exactly 5 builds retained<br>● Unauthorized access blocked<br>● Rollback from stored artifacts works. |
| Regression Testing + Integration Testing + E2E Testing | Deploy change-set + test suite for bug-fix releases. | Trigger pipeline with commit hash + test suite ID → confirm correct selection → run build + test → deploy. | ● Change-set deployed<br>● Automated tests executed<br>● Bug fix traceable via ticket |

SIMPLE

# EPIC 3 — SECURITY, COMPLIANCE & GOVERNANCE

| Type | Acceptance Criterion Being Validated | How It's Verified (Conceptual) | Success Metric / Outcome |
|---|---|---|---|
| Security Testing + RBAC Testing + Audit & Logging Testing | JaaS has RBAC, plugin approvals, security scans, and 90-day audit logs. | Review RBAC config, test access, run plugin approval workflow, verify security scan triggers & audit logs. | ● All controls active<br>● Deprecated plugins flagged<br>● Secure credentials enforced |
| Configuration Compliance Testing + Regression Testing | Jenkins config matches security baseline; Unused/unsafe plugins flagged. | Deploy known CircleCI pipeline into Jenkins and test SCM linkage, Selenium trigger, VM provisioning, and code-check-in automation. | ● All controls active<br>● Deprecated plugins flagged<br>● Secure credentials enforced |
| Access Control Testing + Approval Workflow Testing | Plugin installation must go through controlled approval workflow. | Test authorized vs unauthorized access using RBAC role accounts. | ● No unapproved plugin installed<br>● All requests show audit + sign-off |
| Vulnerability Testing + Security Testing | Vulnerability scans run before plugin installation; High-risk plugins blocked. | Simulate plugin request → go through approval → install → validate plugin availability after restart. | ● Critical/high-risk plugins rejected<br>● Scan results attached to change request. |
| Logging Integrity Testing + Dashboard Access Testing | Audit logs are centralized, tamper-proof, and accessible | Review log entries, timestamps, user IDs → verify dashboard access → confirm 90-day retention | ● Compliance reports available<br>● Logs secure and centralized |
| Integration Testing + Lifecycle Policy Testing | Jenkins pipeline can provision/destroy VMs with defined parameters and lifecycle policy. | Trigger VM creation → verify default config → validate auto-destroy → confirm policy compliance | ● VM created successfully<br>● Auto-destroy works<br>● Configurable parameters respected |

SIMPLE

ISync

# EPIC 4 — THE SENTINEL

| Type | Acceptance Criterion Being Validated | How It's Verified (Conceptual) | Success Metric / Outcome |
|---|---|---|---|
| UAT (User Acceptance Testing) | Persona accurately reflects Security Analyst responsibilities, compliance tasks, and role interactions with DevOps/SRE teams. | Review persona description with security team; validate duties (RBAC checks, audit readiness, plugin approvals) align with real workflows. | • Persona accepted by Security,DevOps, and SRE stakeholders. |
| Security Testing | Persona requires vulnerability scanning before plugin installation and enforcement of risk thresholds. | Simulate plugin install flow → scan → check if high-risk plugins are blocked → confirm approval workflow. | • All high-risk plugins blocked; audit trail attached to request. |
| Integration Testing | Persona requires centralized compliance evidence collection after every build or config change. | Trigger CI pipeline → collect evidence → verify logs/time stamps → validate storage in compliance repository. | • Evidence automatically captured; no missing audit entries. |
| Configuration Regression Testing | Persona requires predefined Jenkins security baselines automatically enforced. | Deploy baseline → run config comparison → detect deviations → flag unsafe or unused plugins. | • Deprecated/unsafe plugins flagged; baseline always matched. |
| Access Control Testing (RBAC Testing) | Persona requires a secure dashboard for accessing audit logs and compliance reports. | Review role-based dashboard access → validate log visibility → test unauthorized access handling. | • Dashboard accessible only to authorized roles; logs correctly displayed. |
| Performance + Integrity Testing | Persona requires 90-day audit log retention, integrity, and immutability. | Validate log rotation policy → review timestamps → test tamper detection → confirm retention window. | • Logs stored immutably for ≥ 90 days; compliance report generated. |

SIMPLE

# BUG MANAGEMENT

## 1. Bug Reporting (Jira Based)

- Centralized Jira board captures all JaaS-related defects（CI/CD flow, RBAC, plugin approvals, VM provisioning), ensuring all issues are tracked in one place.
- Required fields include environment, pipeline ID, logs/screenshots, and clear steps to reproduce so teams can quickly identify root causes.

## 2. Bug Triage & Priority

- Daily triage meeting across App, Security, and Infra teams to review new bugs and assign ownership.
- Issues are marked critical when they block pipelines, bypass RBAC, cause VM provisioning failures, or allow high-risk plugins.

## 3. Bug Resolution & Verification[1]

- Fixes are verified through automated CI/CD reruns, RBAC access retests, and VM create/destroy workflows to confirm stability.
- Critical bugs follow a strict SLA and must be resolved within 48 hours to avoid delays in migration or release.

## 4. Documentation & Traceability

- Every bug links to its related Jenkins job, PR/commit, and relevant Selenium or audit log IDs to maintain full traceability.
- Weekly reporting summarizes defect trends and fix rates, helping teams monitor quality and identify recurring issues.

[1]ISO/IEC/IEEE 29119-4: Software Testing — Part 4: Test Techniques, International Standard for Defect Reporting & Management.

BACKUP

ISync

# APPENDIX

ISync

# TIME BUDGET SUMMARY

| Category | Slides | Time |
|---|---|---|
| BACKUP | 3, 15-17, 19, 21, 36, 45-46, 50, 56, 58, 64-69 | 17 * 0 seconds = 0 minutes |
| TITLE | 1, 2 | 2 * 30 seconds = 1 minutes |
| SIMPLE | 4-6, 7, 9-14, 20, 22-35, 37, 39-44, 47-49, 51-54, 57, 59-63 | 45 * 1 minute = 45 minutes |
| AVERAGE | 8, 38, 55 | 3 * 3 minutes = 9 minutes |
| COMPLEX | 18 | 1 * 5 minutes = 5 minutes |
| TOTAL | | 60 minutes |

# DRAFT REPORT - SPRINT BACKLOG PLAN

## Week 1 & 2 (22 points)

**Oct 5: Project Proposal Context (12 points)**
- Company Mission & Vision(2 points)
- Business Objectives & Hypothesis(4 points)
- Problem Statements & Challenges(4 points)
- Project Scope (2 points)

**Oct 5: OSS Research (5 points)**
- Basic DevOps Research (3 points)
- OSS Decision (2 points)

**Oct 5: OSS Principles (5 points)**
- History (1 points)
- License (1 points)
- Contributors & Sponsors (3 points)

## Week 3 (33 points)

**Oct 12: ROI/ROV (8 points)**
- ROI (4 points)
- ROV (4 points)

**Oct 12: SDLC Model (8 points)**
- Blending (3 points)
- Roadmap & Burnup Chart (5 points)
- Rationalization (3 points)

**Oct 14: OSS Research (Jenkins Focus) (14 points)**
- SWOT Analysis (4 points)
- Competitive Analysis (4 points)
- Upstream & Downstream (6 points)

## Week 4 (29 points)

**Oct 21: Personas & User Stories**
- Clarity & Qualification (3 points)
- Quantification (3 points)
- Decomposition (2 points)
- Rationalization (2 points)

**Oct 21: Risk Management (4 points)**

**Oct 21: Relative Sizing (4 points)**

**Oct 21: SW Disaster (4 points)**

**Oct 23: Reference Work (2 points)**

**Oct 23: Sprint Review Presentation (5 points)**

BACKUP

ISync

# DRAFT REPORT - SPRINT BACKLOG EXECUTION

## Week 1 & 2 (22 points)

**Oct 5: Project Proposal Context (12 points)**

Company Mission & Vision(2 points)
Business Objectives & Hypothesis(4 points)
Problem Statements & Challenges(4 points)
Project Scope (2 points)

**Oct 5: OSS Research (5 points)**

Basic DevOps Research (3 points)
OSS Decision (2 points)

**Oct 5: OSS Principles (5 points)**

History (1 points)
License (1 points)
Contributors & Sponsors (3 points)

## Week 3 (43 points)

**Oct 12: ROI/ROV (8 points)**

ROI (4 points)

ROV (4 points)

**Oct 12: SDLC Model (8 points)**

Blending (3 points)

Roadmap & Burnup Chart (5 points)

Rationalization (3 points)

**Oct 14: OSS Research (Jenkins Focus)**

**(14 points)**

SWOT Analysis (4 points)

Competitive Analysis (4 points)

Upstream & Downstream (6 points)

**Oct 17: Arch & Design (10 points)**

As-Is Diagram (5 points)

To-BE Diagram (5 points)

## Week 4 (29 points)

**Oct 21: Personas & User Stories**

Clarity & Qualification (3 points)

Quantification (3 points)

Decomposition (2 points)

Rationalization (2 points)

**Oct 21: Risk Management (4 points)**

**Oct 21: Relative Sizing (4 points)**

**Oct 21: SW Disaster (4 points)**

**Oct 23: Reference Work (2 points)**

**Oct 23: Sprint Review Presentation**

**(5 points)**

ISync

# FINAL REPORT - SPRINT BACKLOG PLAN

## Week 5 (8 points)

**Oct 30: Arch & Design**

High Level Design (8 points)

## Week 6 (17 points)

**Nov 6: Release Management**

Release Roadmap (6 points)
Release Criteria (5 points)
Release Checklist (6 points)

## Week 7 (16 points)

**Nov 13: Testing**

Testing Strategy (4 points)

Testing Approach (3 points)

Testing Metrics (5 points)

Success Factors & Metrics (4 points)

## Week 8 (12 points)

**Nov 20: Reference Work (2 points)**

**Nov 20: Style & Messaging (10 points)**

Consistent & Clear Style (4 points)

Consistent Messaging (4 points)

Organization (5 points)

## Week 10 (14 points)

**Dec 4 : Pitch Competition**

Pitch Competition Slides (6 points)

OSS Pitch Competition (8 points)

BACKUP

# FINAL REPORT - SPRINT BACKLOG EXECUTION

## Week 5 (16 points)

**Oct 30: Arch & Design**

High Level Design (8 points)

Low Level Design (8 points)

## Week 6 (23 points)

**Nov 6: Release Management**

Release Roadmap (6 points)
Release Criteria (5 points)
Release Checklist (6 points)

**Nov 6: Sprint Planning (6 points)**

## Week 7 (16 points)

**Nov 13: Testing**

Testing Strategy (4 points)

Testing Approach (3 points)

Testing Metrics (5 points)

Success Factors & Metrics (4 points)

## Week 8 (12 points)

**Nov 20: Reference Work (2 points)**

**Nov 20: Style & Messaging (10 points)**

Consistent & Clear Style (4 points)

Consistent Messaging (4 points)

Organization (5 points)

## Week 10 (14 points)

**Dec 4 : Pitch Competition**

Pitch Competition Slides (6 points)

OSS Pitch Competition (8 points)

BACKUP

# CHANGE MANAGEMENT

| Step | Detail |
|---|---|
| **Demonstrate Reasons for the Change** | Clearly articulate why the shift to Jenkins-as-a-Service is needed: inconsistent plugin governance, fragmented pipelines across teams, weak security baselines, high operational overhead in CircleCI, and lack of audit logging. Present benefits such as centralized governance, security hardening, and scalable automation. Define what processes, branches, and CI/CD workflows will be impacted. |
| **Communicate with Stakeholders** | Identify all key stakeholders: DevOps team, pilot teams (M1 & M2), InfoSec, Platform Ops, and leadership. Conduct kick-off meetings, monthly milestone reviews, and demo sessions after each release. Provide continuous updates on rollout progress (M1→M2→M3→M4). Use Slack channels and shared Confluence pages for transparency. |
| **Training & Skill Development** | Provide Jenkins onboarding workshops for pilot teams, including RBAC usage, shared library usage, rollback workflow, and compliance dashboard training. Offer hands-on sandbox sessions for pipeline migration and plugin governance approvals. Ensure developers understand security baselines and audit logging. |
| **New Talent Acquisition** | Recruit or assign Jenkins SMEs to support shared library maintenance and plugin governance. Add 1–2 DevOps engineers to support enterprise rollout in M4. Engage InfoSec analysts to maintain compliance baselines and perform CVSS evaluations on Jenkins plugins. |
| **Evaluation, Feedback & Documentation** | After each release (v1.0→v1.1→v1.2→v1.3), collect feedback from pilot teams and DevOps. Evaluate expected vs actual stability, build rate, compliance readiness, and operational issues. Maintain documentation in Confluence: migration guides, plugin governance rules, RBAC matrix, audit log configuration, rollback runbook. |
| **Reinforcement** | Provide ongoing operational support for Jenkins, including plugin reviews, shared library updates, RBAC enforcement, and periodic compliance checks. Run lessons-learned retros after each release to reinforce best practices. |