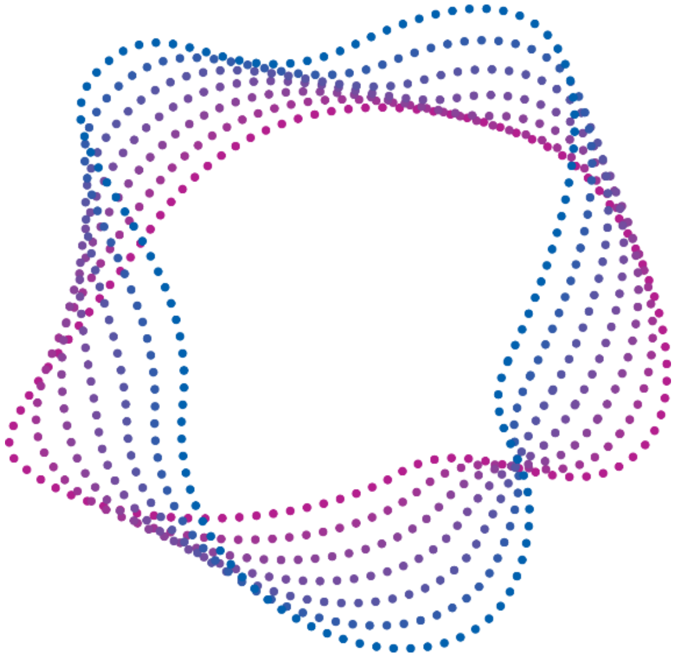
Enhancing DevSecOps Maturity with Metrics

SHIP-HATS Learning Events

May 2024

Agenda



- ➢Engineering Productivity Programme (EPP) and Recap of DevSecOps Metrix

  Liyana Muhammad Fauzi, Lead Product Manager, GDS, GovTech

- ➢Understanding your Metrics

  Kelvin Leong, Cybersecurity Engineer, GDS, GovTech

- ➢Improving Deployment Frequency

  Leon Leow, Product Manager, GDS, GovTech

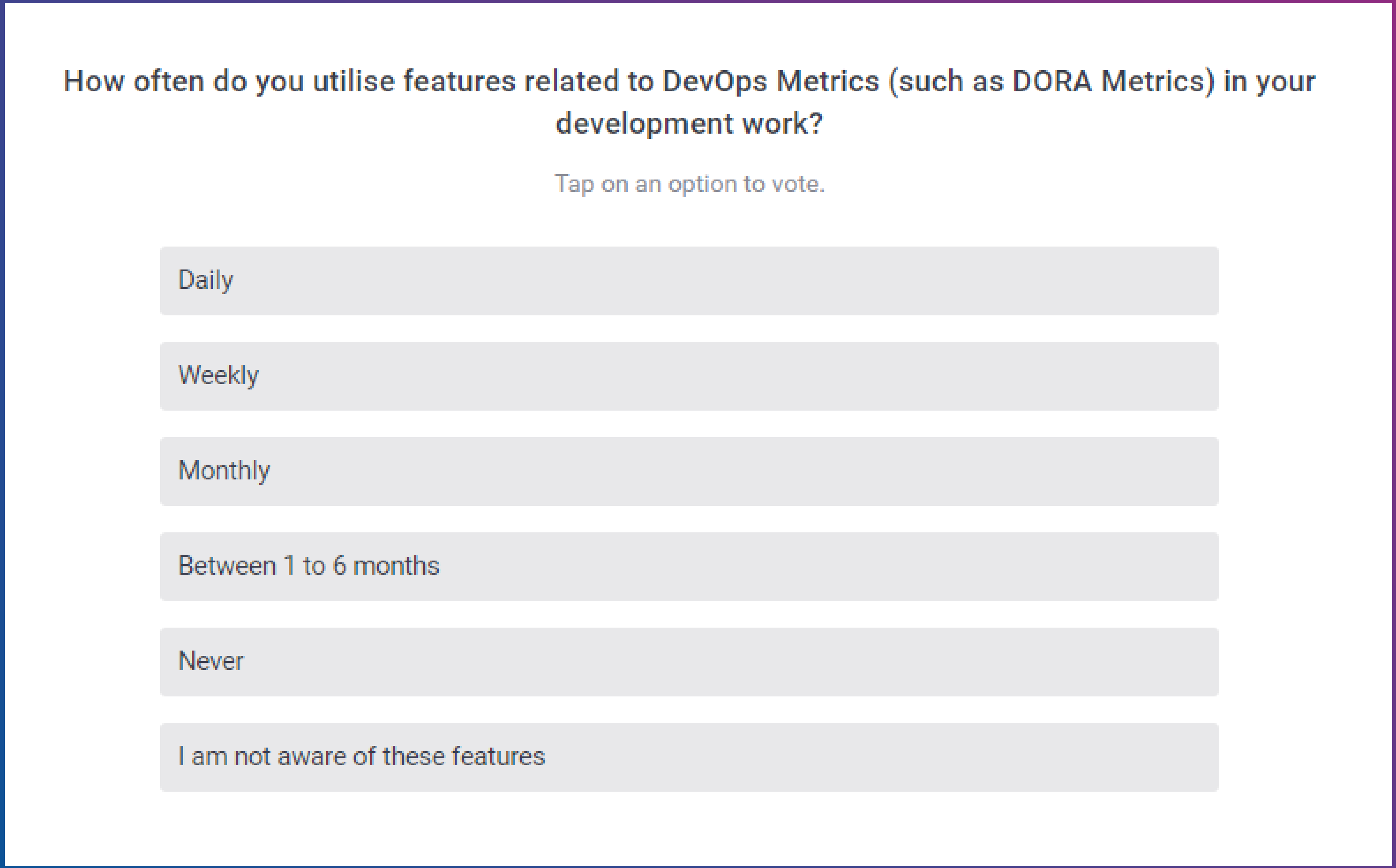
- ➢Next steps

  Leon Leow, Product Manager, GDS, GovTech

  Hudson Lee, Principal DevOps Engineer, GDS, GovTech

Poll

Scan the code and answer the poll!



How often do you utilise features related to DevOps Metrics (such as DORA Metrics) in your development work?

Tap on an option to vote.

- Daily
- Weekly
- Monthly
- Between 1 to 6 months
- Never
- I am not aware of these features

https://pigeonhole.at/SHIPHATS

Engineering Productivity Programme (EPP)

Liyana Muhammad Fauzi

shiphats

What is the Engineering Productivity Programme?

Optimise government software engineering productivity and developer experience, and enable agencies to deliver and operate, reliable, compliant and cost-effective digital products efficiently.

Optimise Cost

Reduce time to market

Increase software re-use

Improve Quality and Resiliency

Accelerating Engineering Productivity

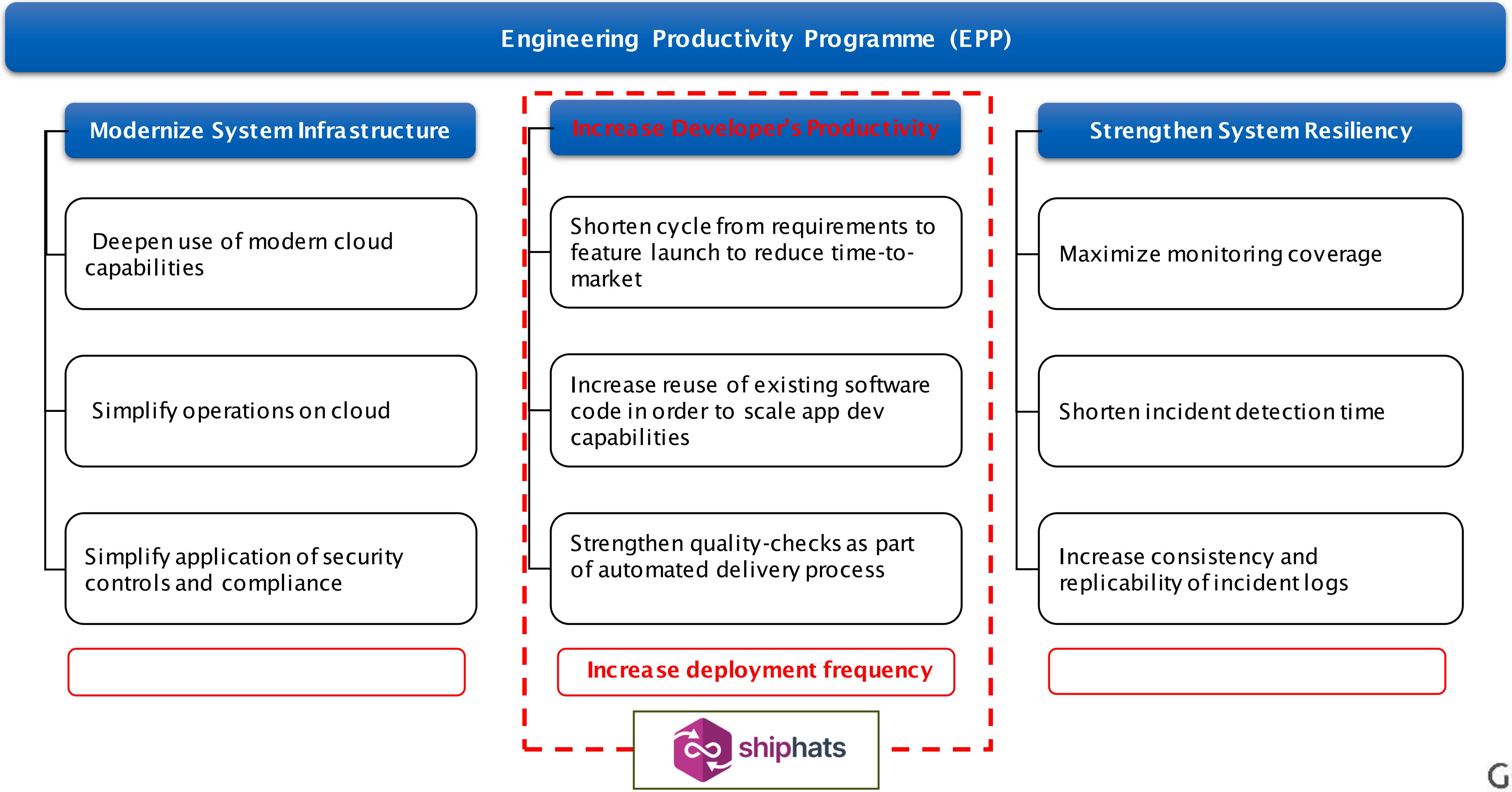
EPP Goals

- Deepen application of modern application architecture
- Realize benefits from end-to-end developer experience
- Improve system resiliency through performance management

Beneficiaries

- Whole-of-Government (WOG) developers from different Government Agencies
- Vendors that are working on Government Projects

Outcome-Driven Sub-Programmes



Engineering Productivity Programme (EPP)

Modernize System Infrastructure

- Deepen use of modern cloud capabilities
- Simplify operations on cloud
- Simplify application of security controls and compliance

Increase Developer's Productivity

- Shorten cycle from requirements to feature launch to reduce time-to-market
- Increase reuse of existing software code in order to scale app dev capabilities
- Strengthen quality-checks as part of automated delivery process

Increase deployment frequency

shiphats

Strengthen System Resiliency

- Maximize monitoring coverage
- Shorten incident detection time
- Increase consistency and replicability of incident logs

SHIP-HATS – Source Code Mgmt and CICD Platform

shiphats

Access (by default)

Subscription Mgmt

techbiz

Tools Provisioning

SHIP-HATS Portal

Sign in

techpass

SEED

Plan & Code

Build

Build Testing
SAST

Other Tests
Code Quality | Container Scan | DAST

Deploy & Release

Main Toolchain

GitLab
The One DevOps Platform

GitLab Native

Covers E2E SCM, CI/CD, open source & vulnerabilities scan, project management and metrics.
Sufficient coverage in all areas but may not be best-in-class
Alternative tools below are best-in-class for agencies with higher requirements

Alternative Tools

Confluence
Jira

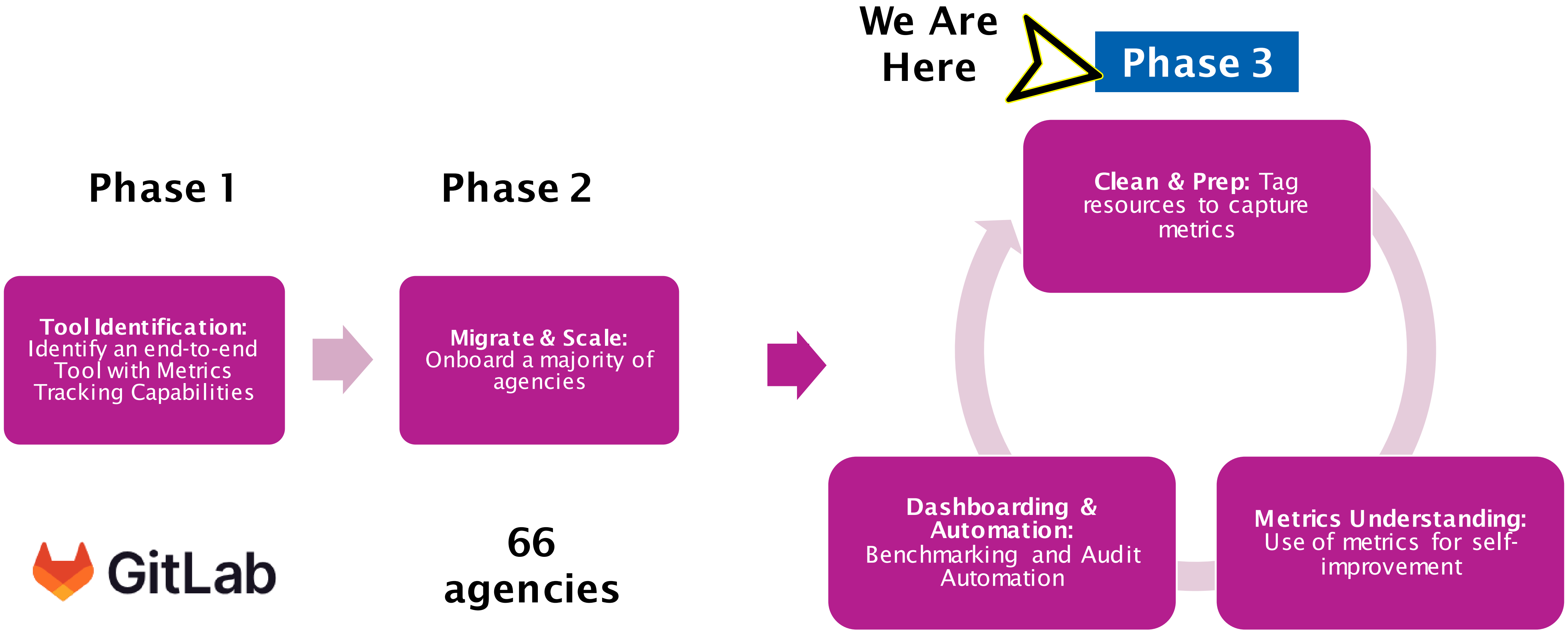
nexus repository pro
sonatype lifecycle

Fortify on Demand
sonarqube
pCloudy.com

Coding Assistants (New)

GOVTECH SINGAPORE

Capturing DevSecOps Metrics



Phase 1

Tool Identification: Identify an end-to-end Tool with Metrics Tracking Capabilities

GitLab

Phase 2

Migrate & Scale: Onboard a majority of agencies

66 agencies

We Are Here

Phase 3

Clean & Prep: Tag resources to capture metrics

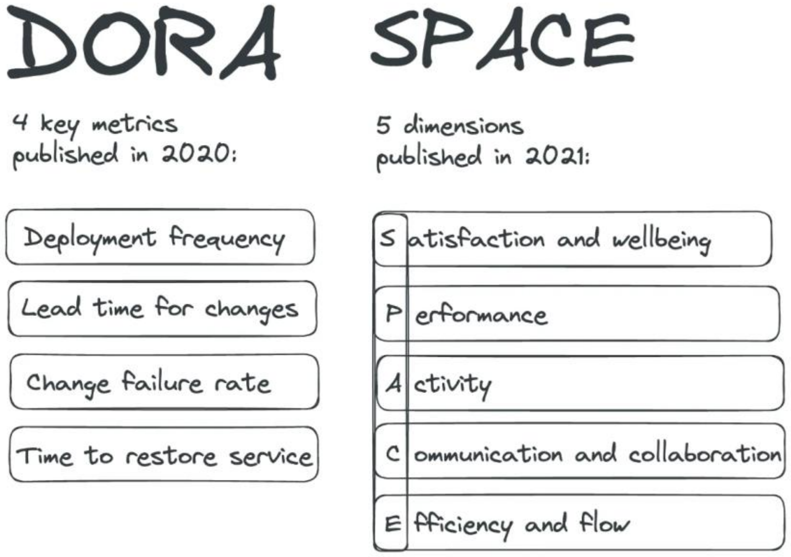
Metrics Understanding: Use of metrics for self-improvement

Dashboarding & Automation: Benchmarking and Audit Automation

DevSecOps Measurements: Many Definitions

DORA

DEVOPS RESEARCH & ASSESSMENT



DORA

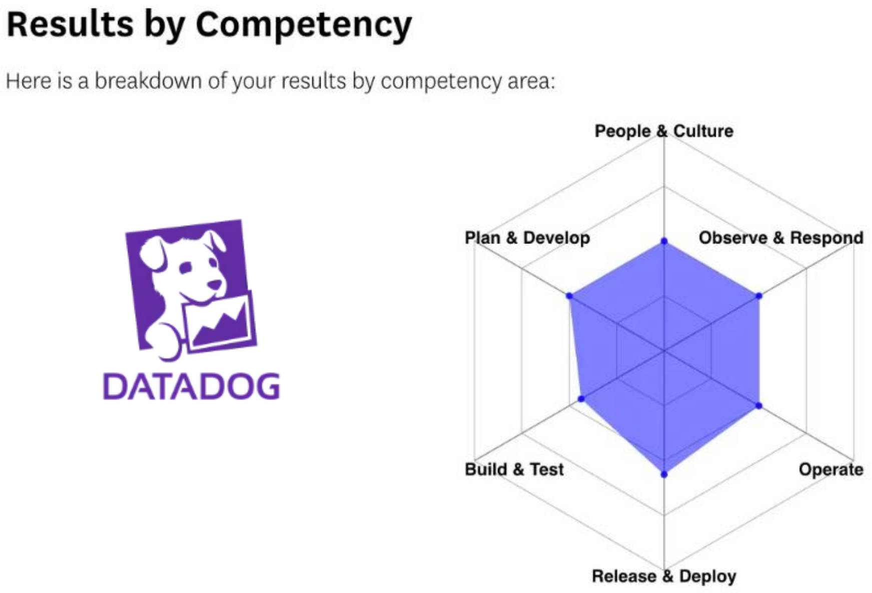
4 key metrics
published in 2020:

- Deployment frequency
- Lead time for changes
- Change failure rate
- Time to restore service

SPACE

5 dimensions
published in 2021:

- S atisfaction and wellbeing
- P erformance
- A ctivity
- C ommunication and collaboration
- E fficiency and flow



Results by Competency

Here is a breakdown of your results by competency area:

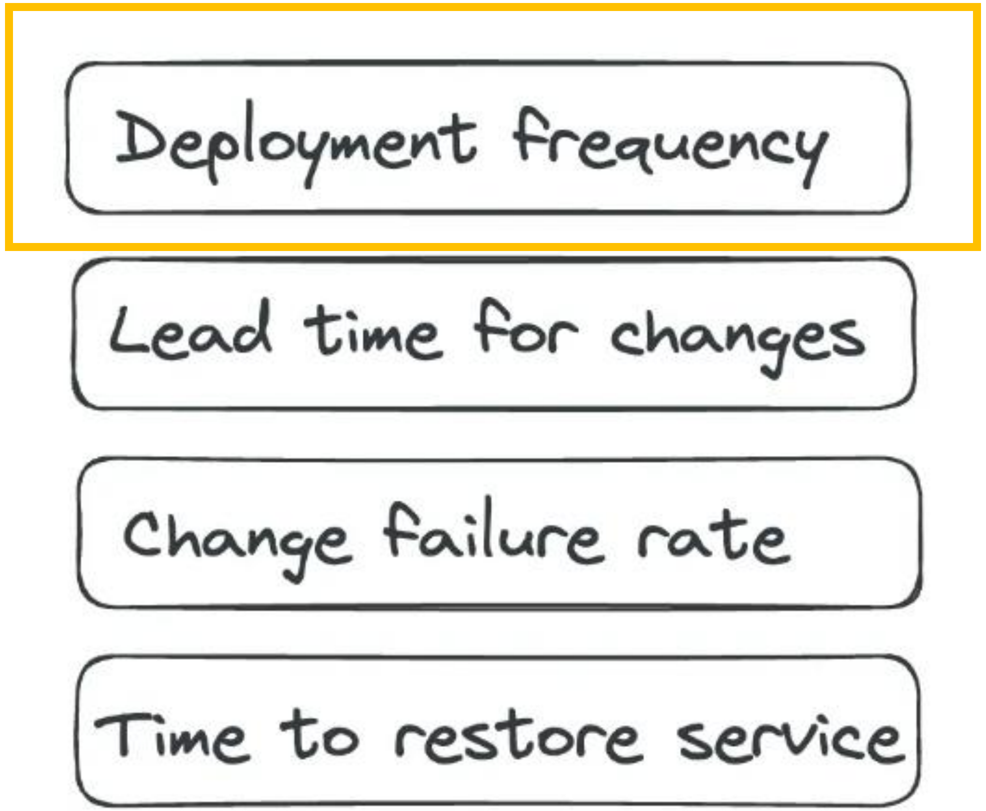
OWASP Devsecops Maturity Model

DSOMM

Recap: DORA

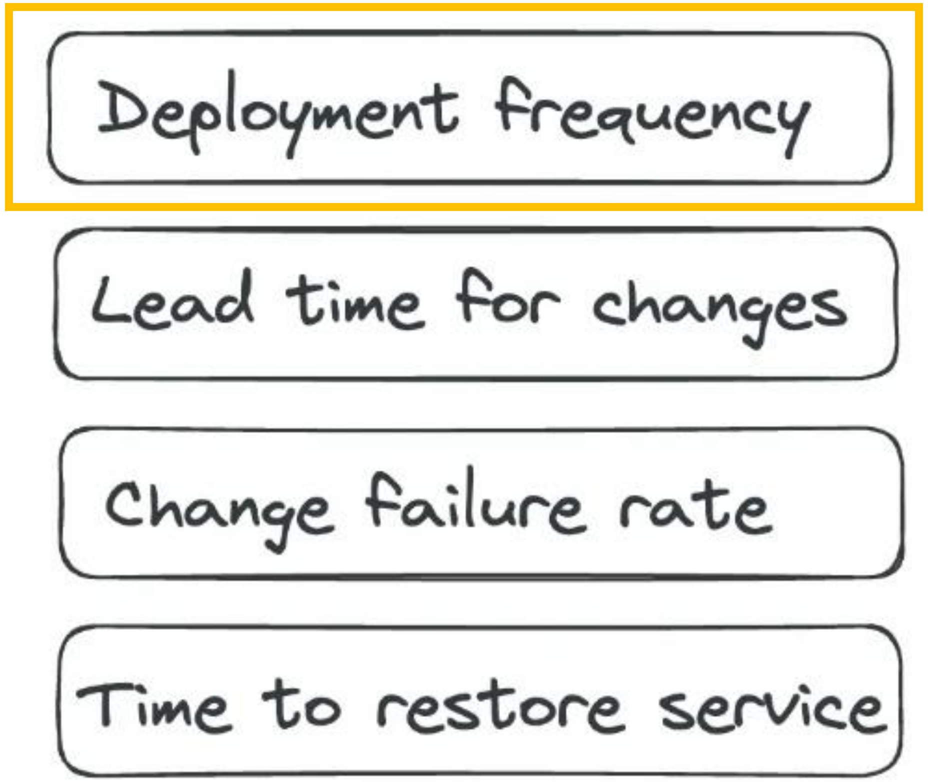
1. DORA - "**DevOps Research and Assessment**“
2. A **research program** that was founded in 2014 by Dr. Nicole Forsgren, Jez Humble, and Gene Kim
3. **State of DevOps report** provides insights on high-performing organizations.
4. **DevOps Assessment tool** identifies areas for improvement.
5. **High-performing IT organizations achieve business goals**

Priority



What do the DORA metrics represent?

Priority



Deployment frequency

Lead time for changes

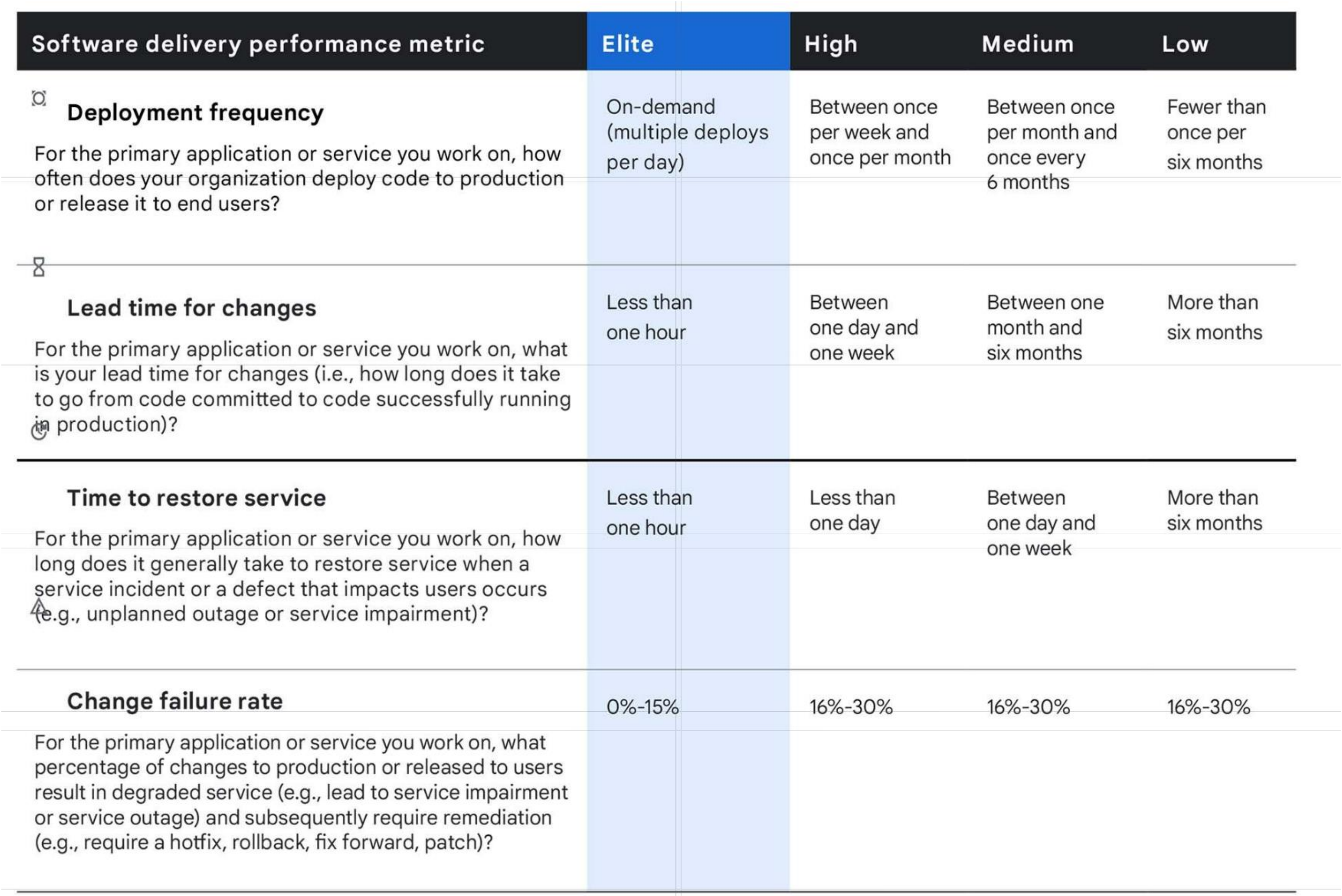
Change failure rate

Time to restore service

a. **Efficiency**
   1. **Deployment Frequency:** How often an organization deploys code to production.
   2. **Lead Time for Changes:** The time it takes for a change to go from code commit to production.

b. **Quality**
   1. **Change Failure Rate:** The percentage of deployments that fail in production.
   2. **Time to Restore Service:** How quickly a team can recover from a failure in production.

Industry Benchmarks vs SG Gov Realities

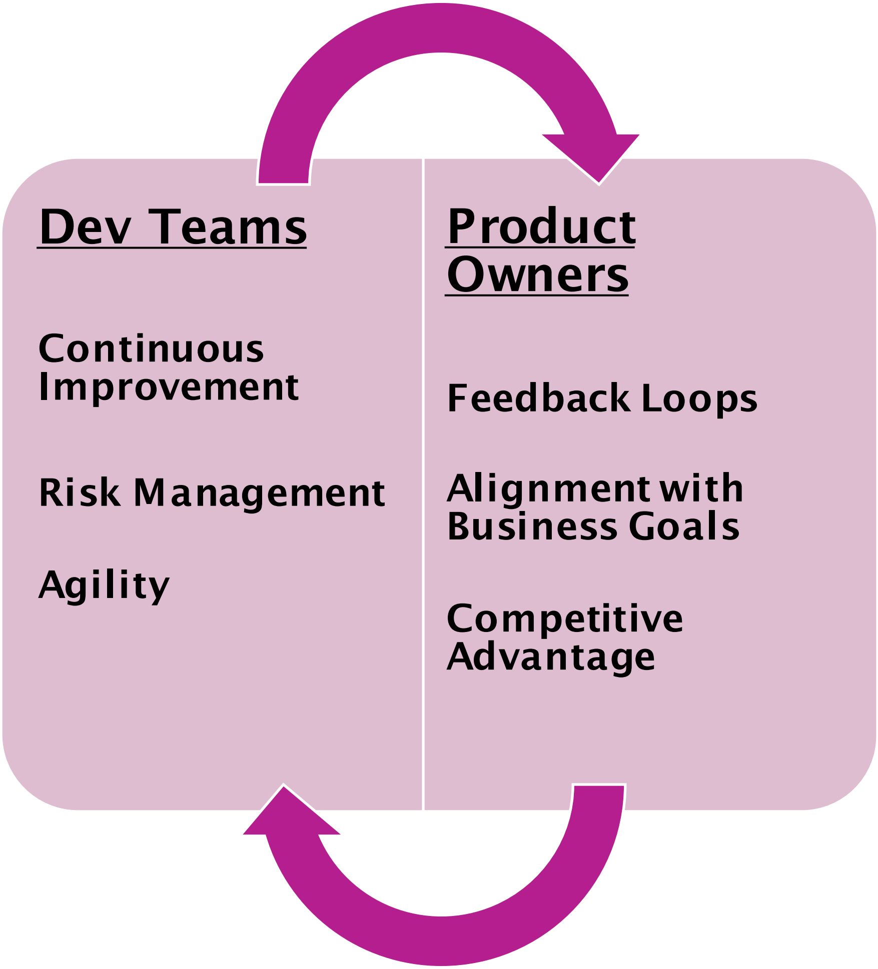


Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

- Do government practices enable us to benchmark against industry norms?
- What are some peculiarities in the gov construct that might limit us?

Why start with Deployment Frequency?

- Encourages the establishment of a robust CICD pipeline
- Faster Feedback Loop
- Reduced Risks
- Enhanced Quality
- Increased Agility



Dev Teams

Continuous Improvement

Risk Management

Agility

Product Owners

Feedback Loops

Alignment with Business Goals

Competitive Advantage

For Dev Teams: Deployment Frequency as a metric

Dev Teams

Continuous Improvement

Risk Management

Agility

Continuous Improvement
Identify trends and make improvements in development and release processes
Measure how often new code or updates are deployed

Risk Management
Insights into the stability of the deployment process

Agility
Higher deployment frequency reflects an agile development approach
Allows teams to respond quickly to market demands and customer feedback

For Product Owners: Deployment Frequency as a metric

Feedback Loops
Crucial for product iterations and improvements
Frequent deployments enable faster feedback loops with users

Alignment with Business Goals
Align efforts with business objectives
Ensure timely release of features and fixes

Competitive Advantage
Ability to deploy frequently can be a competitive advantage
Stay ahead in the fastpaced tech industry

Product Owners

Feedback Loops

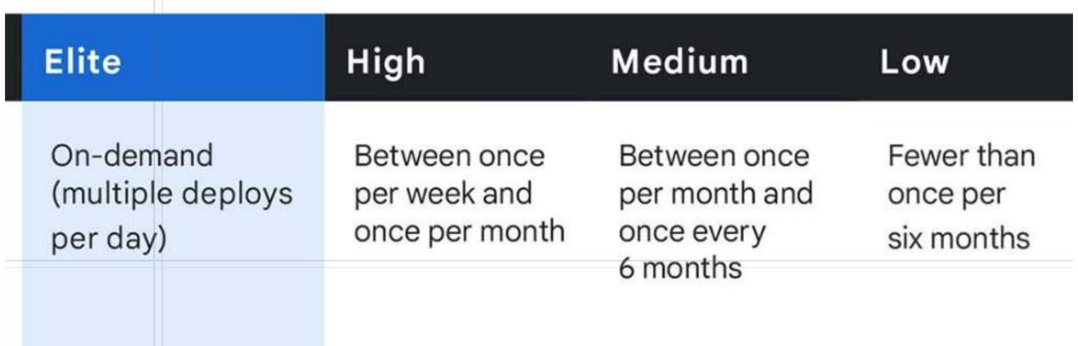
Alignment with Business Goals

Competitive Advantage

Deployment Frequency in SHIP-HATS Today

- 500+ systems, across 66 agencies
- Majority are not tagged to production resulting in 0 deployments

	Elite	High	Med	Low	0
Systems (over the past 90 days)	6	8	57	86	**396**



Elite	High	Medium	Low
On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months

3 Key Takeaways

1. Engineering Productivity is core to delivering quality and secure applications
2. DORA metrics give us some insight into the efficiency and the quality of the outcomes from product and development teams
3. Deployment frequency is the starting point for having conversations about how product and development teams can do better at delivering quality and secure applications

Q & A

Scan the code and add questions!

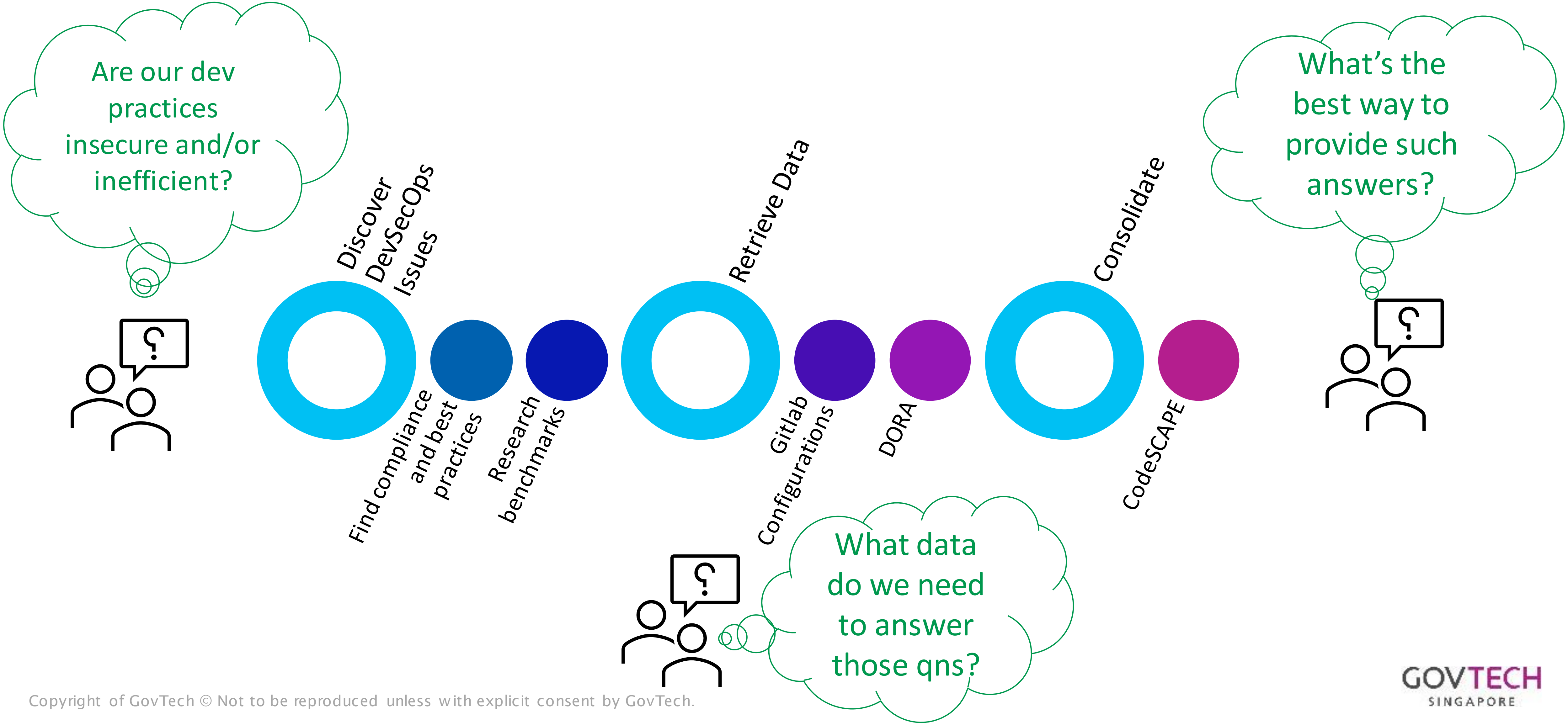
https://pigeonhole.at/SHIPHATS

Understanding your Metrics

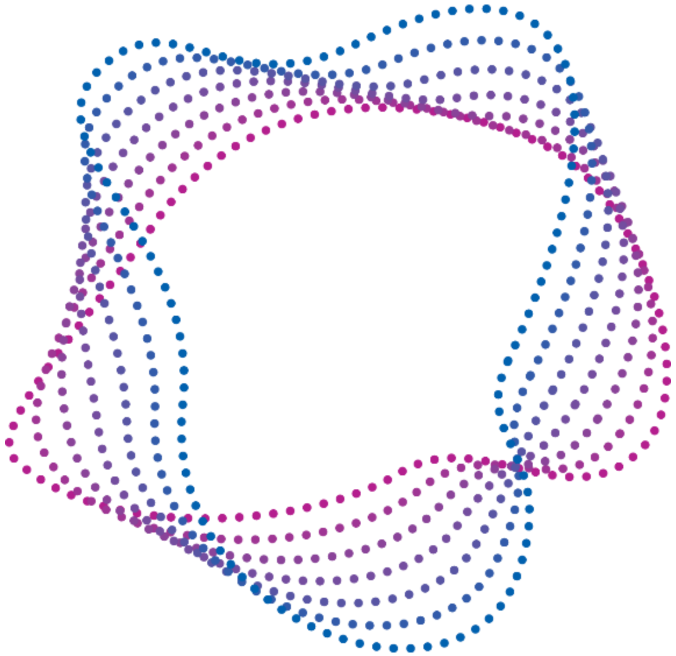
Kelvin Leong

CodeSCAPE - Recap

Providing oversight on DevSecOps practices and insights on DORA


Are our dev practices insecure and/or inefficient?
Discover DevSecOps Issues
Find compliance and best practices
Research benchmarks
Retrieve Data
Gitlab Configurations
DORA
What data do we need to answer those qns?
Consolidate
CodeSCAPE
What's the best way to provide such answers?

CodeSCAPE Demo

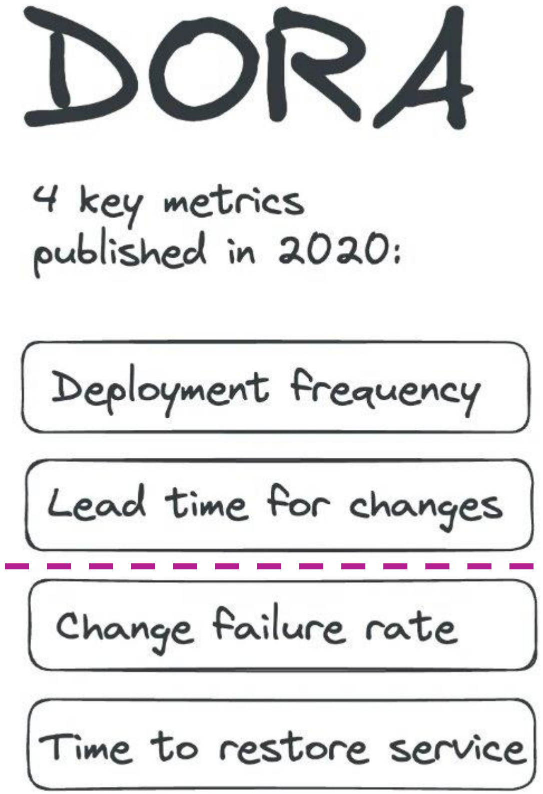


Discover how CodeSCAPE (pilot) can refine your DevSecOps practices

GOVTECH SINGAPORE

What can I do to Implement GitLab DORA?

DORA: Implementation Requirements



Measures development performance (velocity)
Days to implement: ~5 days

Measures development quality (stability)
Days to implement: ~3 days

Tag Environments

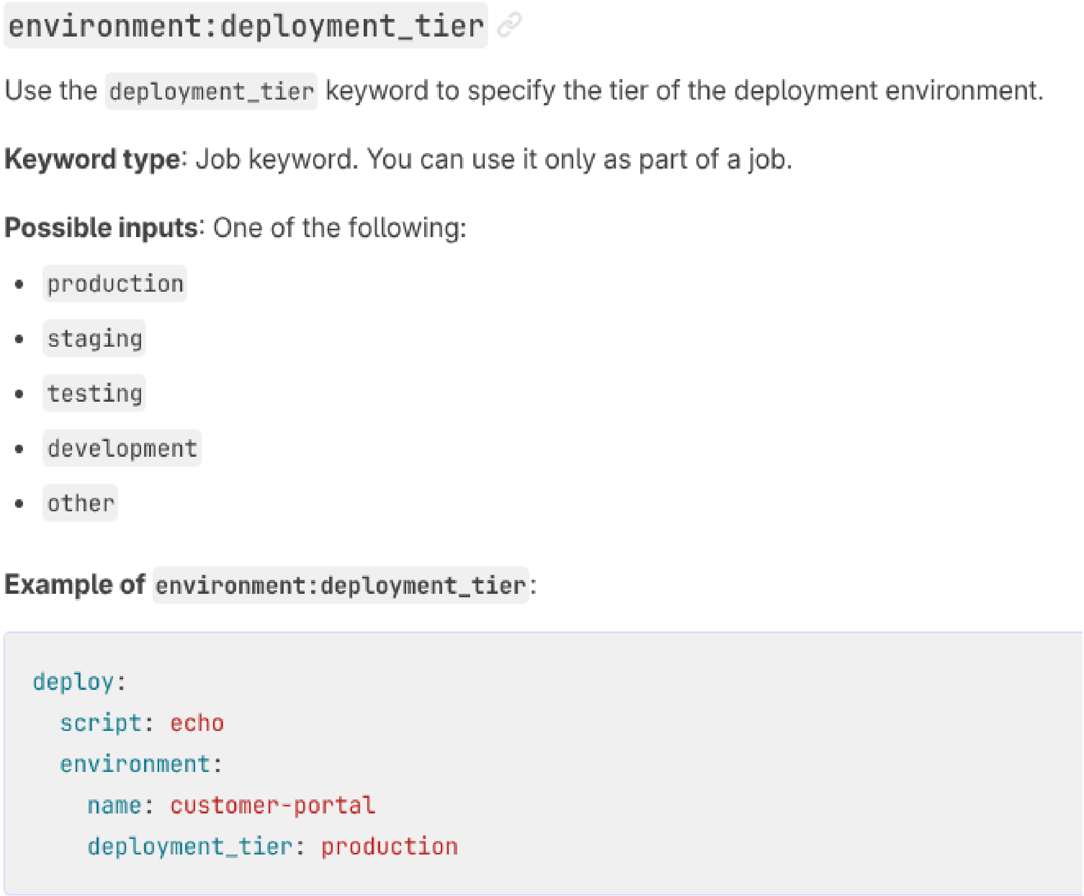
Deployment Record

Merge Request

GitLab Incident

Tagging Environments

- GitLab only considers events that happens to Production environment for DORA calculations.
- There are mainly two ways to tag a production environment:
  - Name your environment as **production** or **prod** in project settings or your pipeline job
    - https://docs.gitlab.com/ee/ci/yaml/index.html#environmentname
  - Use the “deployment_tier” variable to mark an environment as production tier
    - https://docs.gitlab.com/ee/ci/yaml/index.html#environmentdeployment_tier



`environment:deployment_tier`

Use the `deployment_tier` keyword to specify the tier of the deployment environment.

Keyword type: Job keyword. You can use it only as part of a job.

Possible inputs: One of the following:

- `production`
- `staging`
- `testing`
- `development`
- `other`

Example of `environment:deployment_tier`:

```yaml
deploy:
  script: echo
  environment:
    name: customer-portal
    deployment_tier: production
```

Tagging Environments

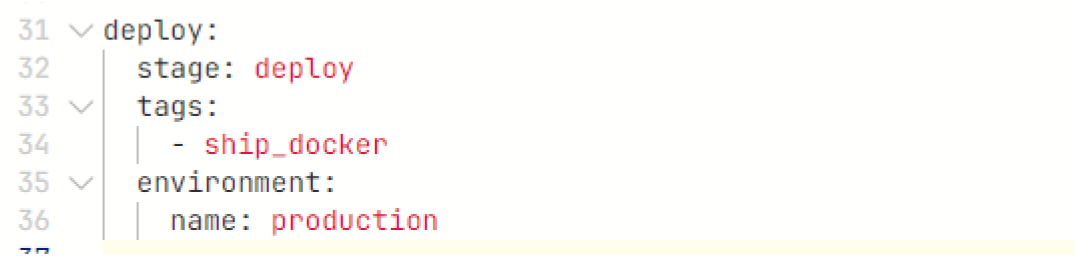
- **Case 1**
  - Project does not have any environment and the team is ok to name the environment as “production”

- **Steps**
  - Either:
    - → Name environment as **production** or **prod** in the pipeline deployment job ①

    OR

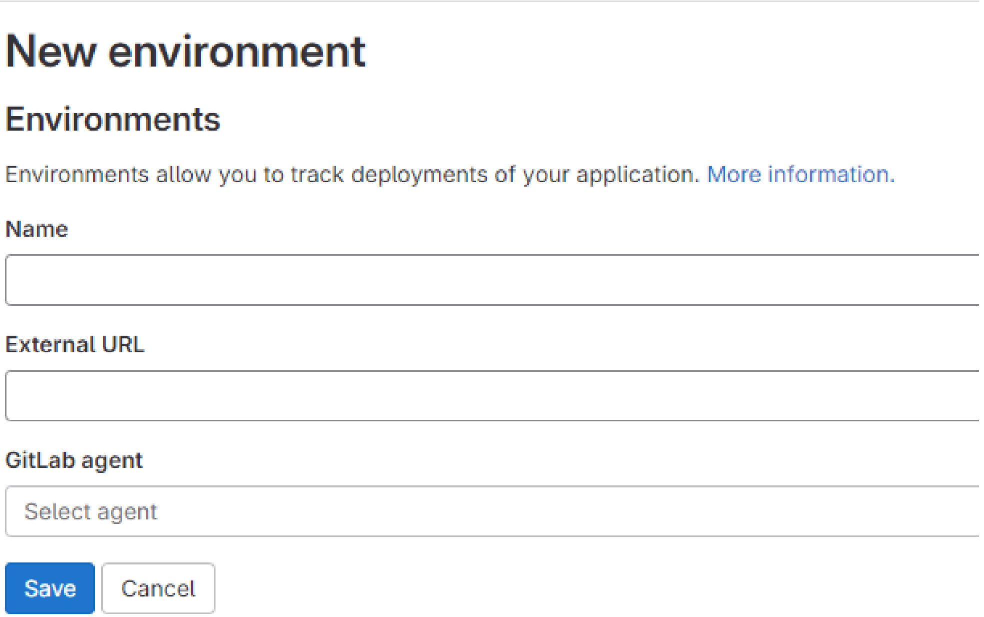
    - → In the projects page, using the left panel, navigate to “Operate” > “Environments”
    - → Click on “New environments” ②
    - → Ensure the “Name” field is filled with **production** or **prod**
    - → Enter the other fields as necessary and click “Save”
  - GitLab will automatically use the name to deduce that it is a production tier environment

①



```
deploy:
  stage: deploy
  tags:
    - ship_docker
  environment:
    name: production
```

②



New environment

Environments

Environments allow you to track deployments of your application. More information.

Name

External URL

GitLab agent

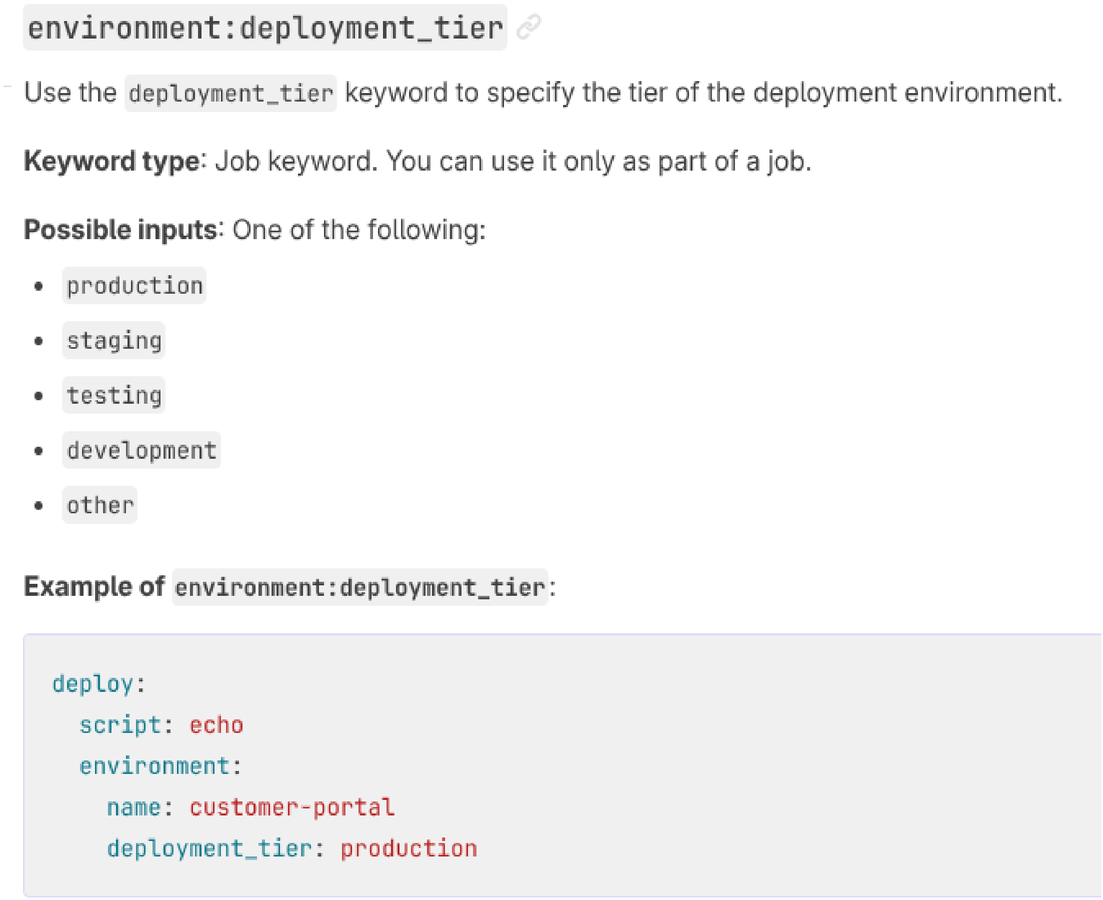
Select agent

Tagging Environments

- **Case 2**
  - Project does not have any environment and the team wants a custom name for the environment but still wishes to tag it as “production” tier

- **Case 3**
  - Project already has an environment

- **Steps (Case 2 and 3)**
  - Use the **“deployment_tier”** variable and set it to **“production”** while creating/updating your environment
    - → The variable is available for use in:
      - **Pipeline deployment job**
        - https://docs.gitlab.com/ee/ci/yaml/index.html#environmentdeployment_tier
      - **Environment creation API**
        - https://docs.gitlab.com/ee/api/environments.html#create-a-new-environment
      - Environment update API
        - https://docs.gitlab.com/ee/api/environments.html#update-an-existing-environment



`environment:deployment_tier`

Use the `deployment_tier` keyword to specify the tier of the deployment environment.

Keyword type: Job keyword. You can use it only as part of a job.

Possible inputs: One of the following:

- `production`
- `staging`
- `testing`
- `development`
- `other`

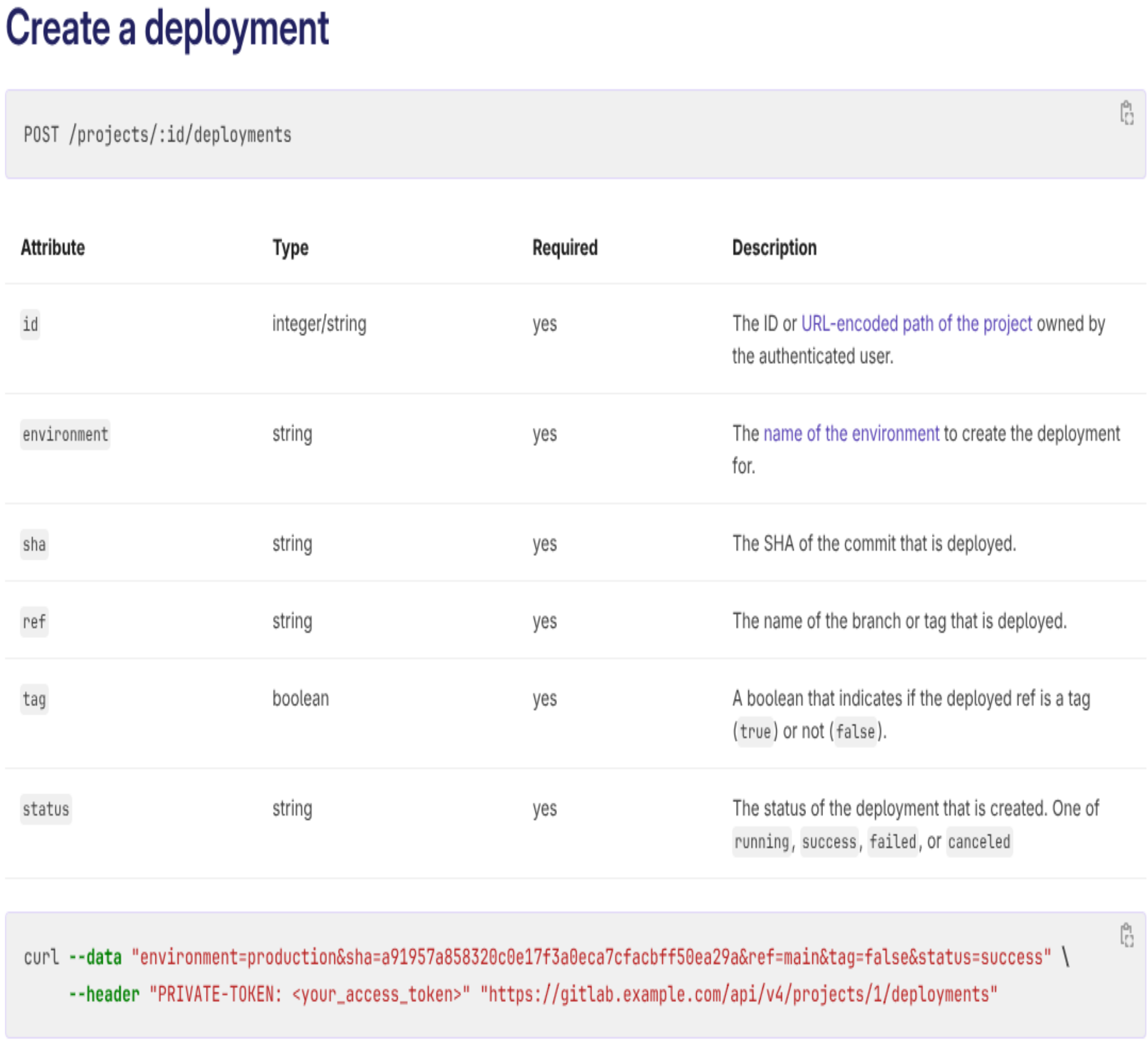
Example of `environment:deployment_tier`:

```yaml
deploy:
  script: echo
  environment:
    name: customer-portal
    deployment_tier: production
```

Deployment records

- There are mainly two ways to get deployments recorded in GitLab
  - Use a CI/CD job for deployment
  - Create a deployment record using the API
    - https://docs.gitlab.com/ee/api/deployments.html#create-a-deployment

Note: DORA only measures deployments to production tier



Create a deployment

```
POST /projects/:id/deployments
```

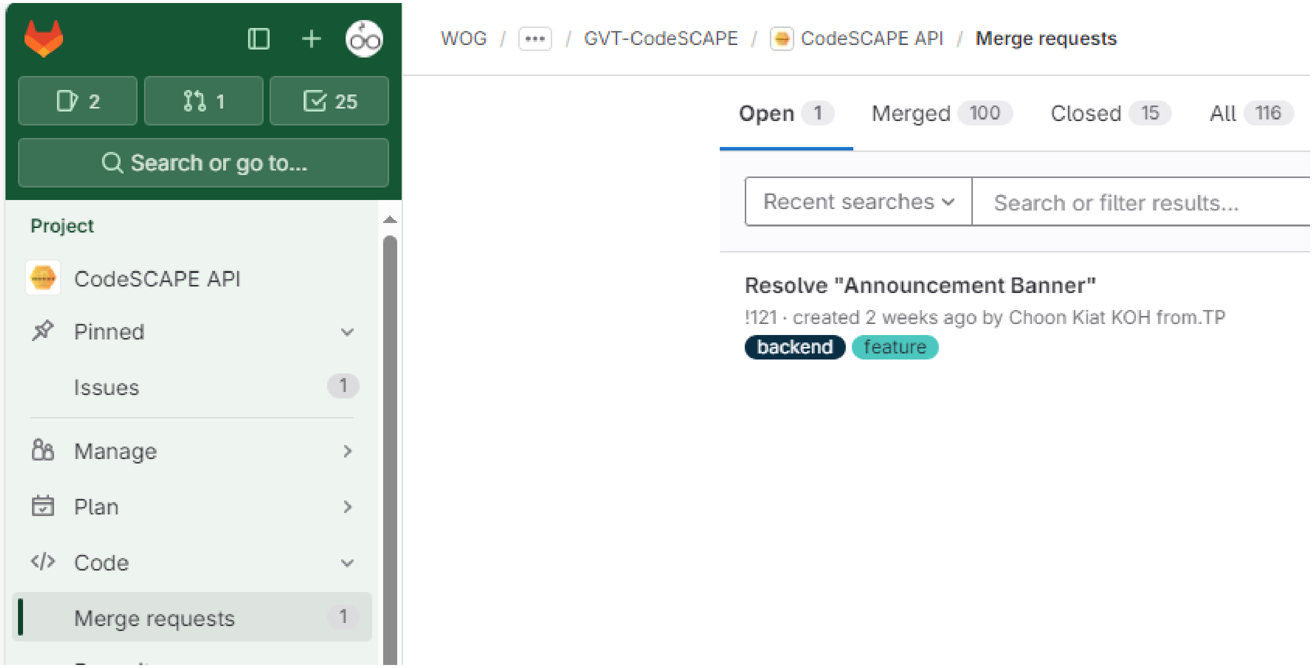
Attribute	Type	Required	Description
`id`	integer/string	yes	The ID or URL-encoded path of the project owned by the authenticated user.
`environment`	string	yes	The name of the environment to create the deployment for.
`sha`	string	yes	The SHA of the commit that is deployed.
`ref`	string	yes	The name of the branch or tag that is deployed.
`tag`	boolean	yes	A boolean that indicates if the deployed ref is a tag (`true`) or not (`false`).
`status`	string	yes	The status of the deployment that is created. One of `running`, `success`, `failed`, or `canceled`

```
curl --data "environment=production&sha=a91957a858320c0e17f3a0eca7cfacbff50ea29a&ref=main&tag=false&status=success" \
    --header "PRIVATE-TOKEN: <your_access_token>" "https://gitlab.example.com/api/v4/projects/1/deployments"
```

Using Merge Requests

- The merge request feature will help to track the mean time for change.

- This assumes the project does not make direct code changes to branches that effects a deployment to production environment
  - https://docs.gitlab.com/ee/user/project/merge_requests/



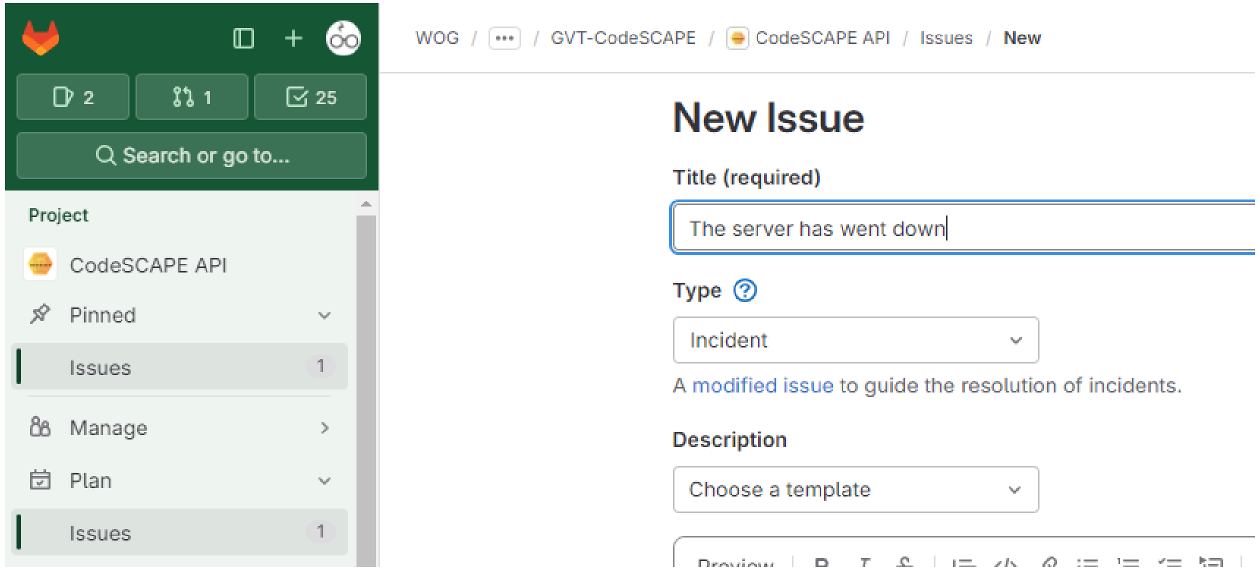
Resolve "Announcement Banner"
!121 · created 2 weeks ago by Choon Kiat KOH from.TP
backend feature

Using GitLab Incidents

- GitLab uses Incidents feature for
  - Change Failure Rate
  - Time to Restore Service

- GitLab assumes all incidents pertains to production tier

- Incidents can be created both manually and automatically
  - https://docs.gitlab.com/ee/operations/incident_management/incidents.html
  - https://docs.gitlab.com/ee/operations/incident_management/manage_incidents.html#create-an-incident



WOG / ••• / GVT-CodeSCAPE / CodeSCAPE API / Issues / New

New Issue

Title (required)

The server has went down

Type

Incident

A modified issue to guide the resolution of incidents.

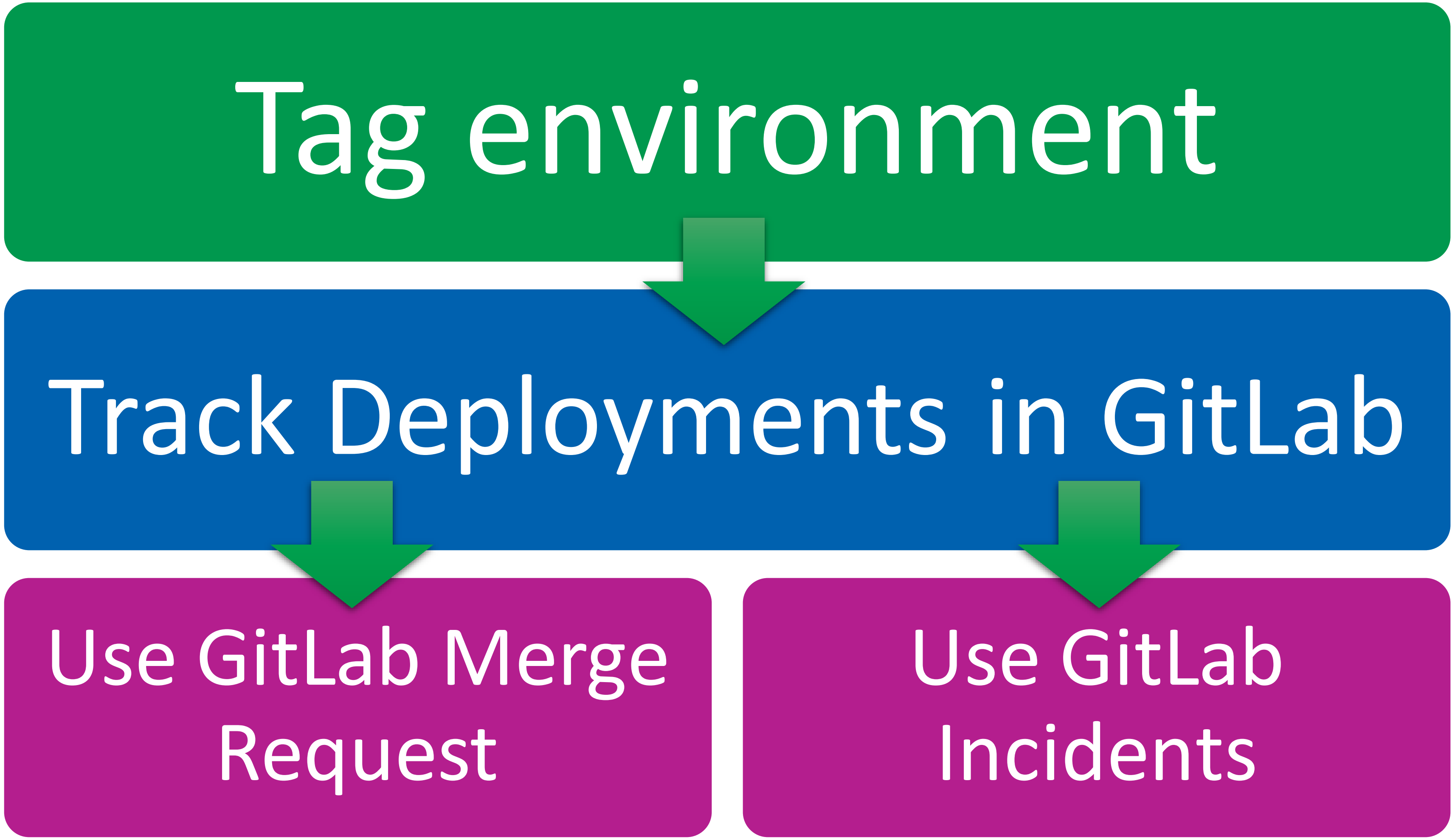
Description

Choose a template

Caveats and Challenges

- DORA data might have some **lag time** due to manual or delayed data population
- GitLab's DORA calculation requires **data to be within GitLab**.
  - We are currently looking into tools to enable integration with other software (e.g., Jira)
- Network segregated systems might need to build **custom workflows**, and/or middleware to accurately populate the required data (e.g., deployment data) if needed.

DORA: Implementation Summary



Tag environment

Track Deployments in GitLab

Use GitLab Merge Request

Use GitLab Incidents

CodeSCAPE Pilot Registration

- Sign up for pilot access to start your DevSecOps refining journey now!
  - https://go.gov.sg/codescape-pilot

- CodeSCAPE Documentation:
  - https://go.gov.sg/codescape

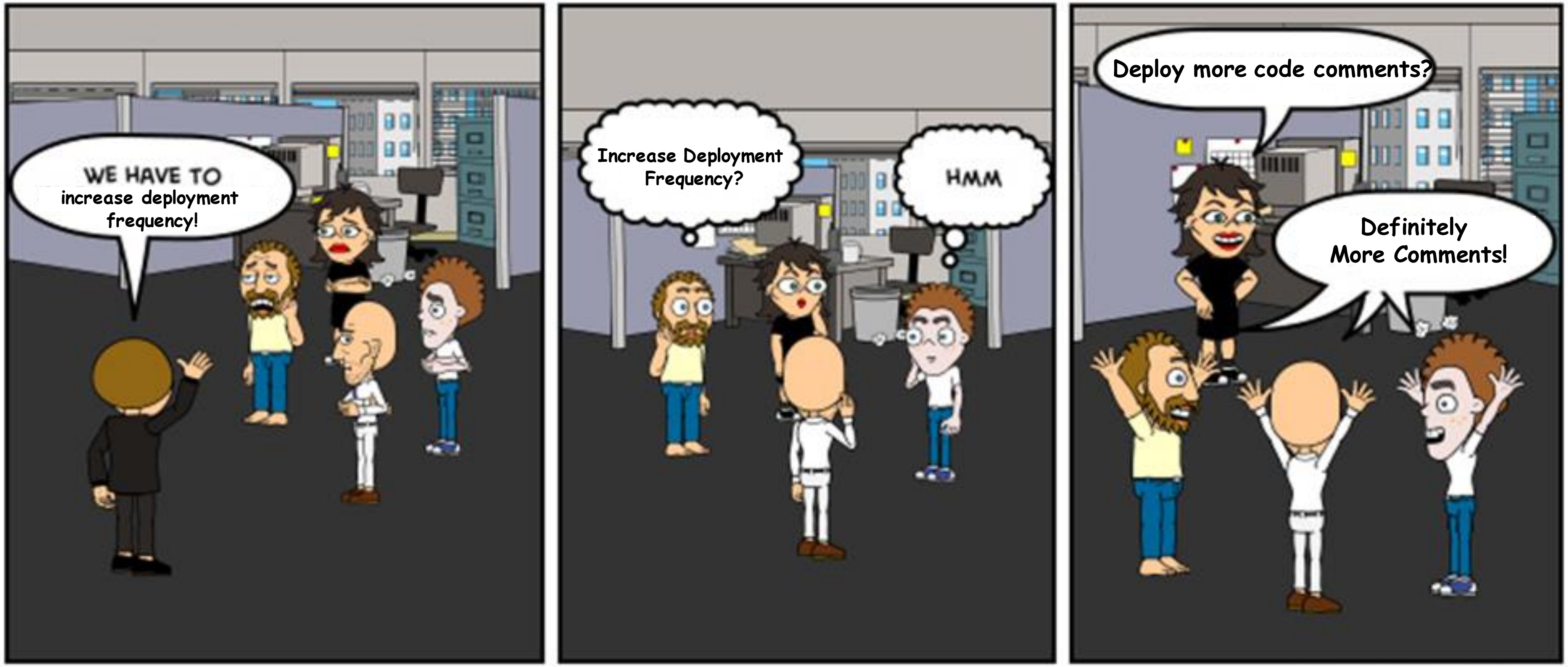
https://go.gov.sg/codescape-pilot

Improving Deployment Frequency

Leon Leow

What you should not do :)

Gaming Deployment Frequency

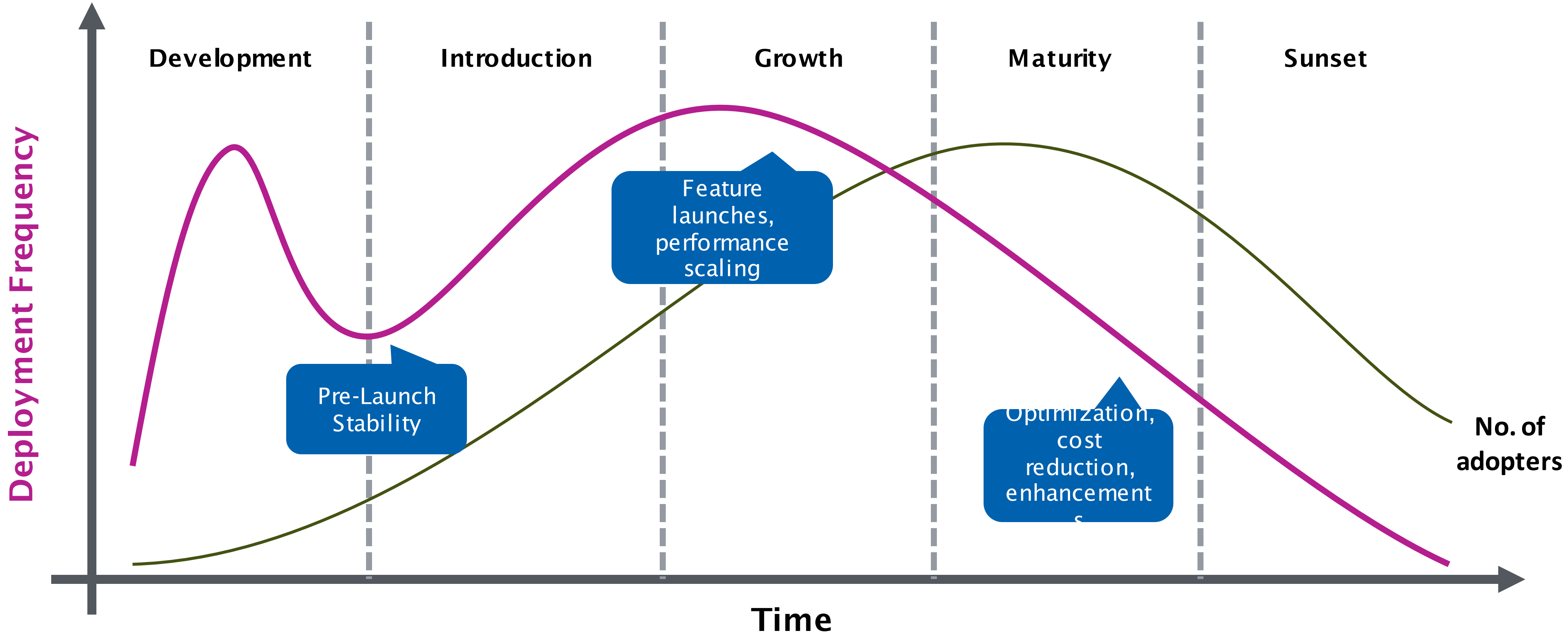

WE HAVE TO increase deployment frequency!
Increase Deployment Frequency?
HMM
Deploy more code comments?
Definitely More Comments!

Adapted from AXOSOFT.COM

WWW.BITSTRIPS.COM

"The practices that would be shared are not tooling specific"

#0: Deployment Frequency fluctuates over a product’s lifecycle



Measuring a team’s capability to frequently deploy is more important than to aimlessly increase it

#1: You cannot improve what you cannot measure

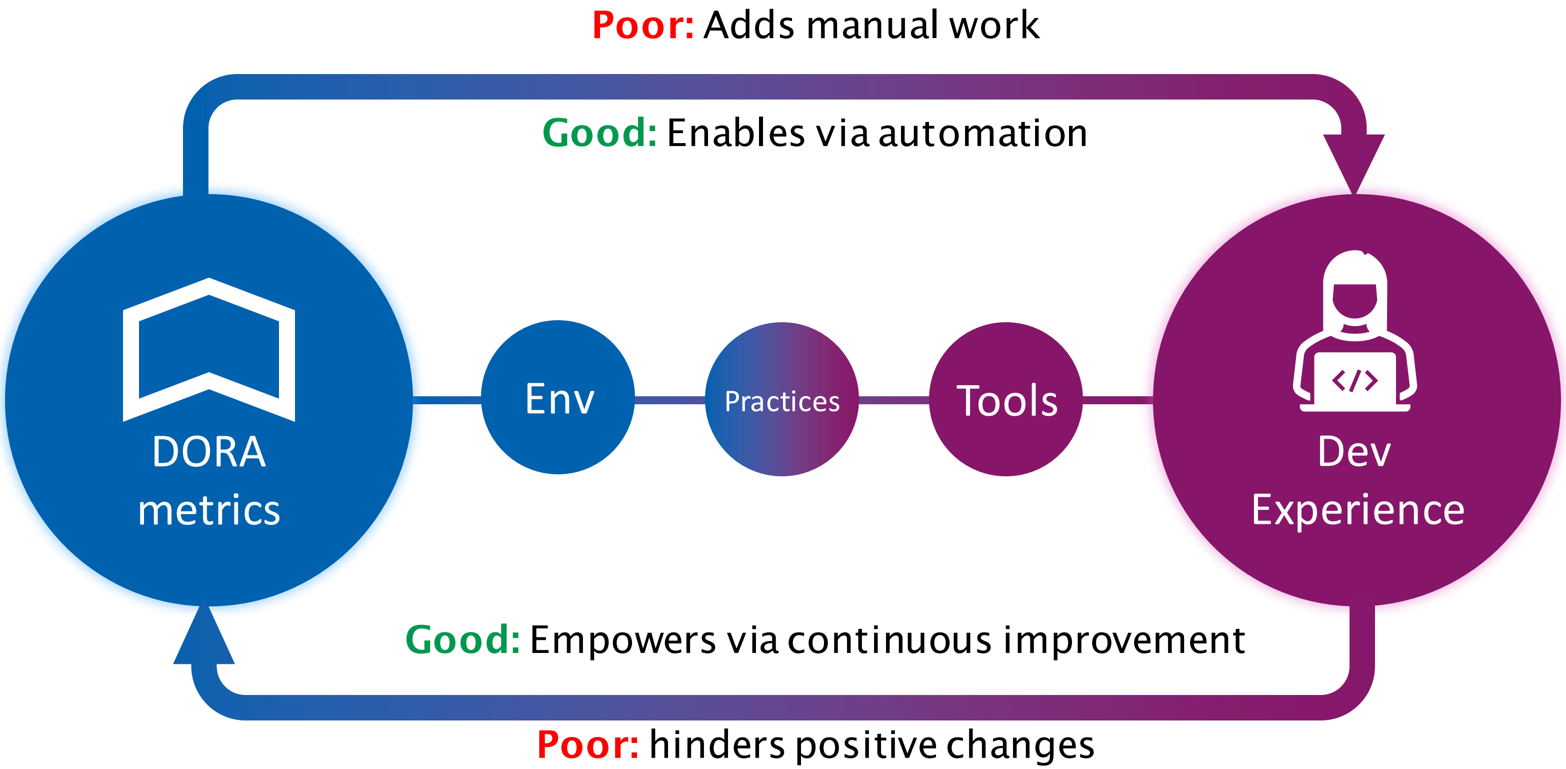
Define	A realistic deployment frequency considering lifecycle of product
Measure	The current baseline of deployment frequency
Analyse	Why is deployment frequency not at the expected levels today
Improve	By designing, executing experiments and validating against baseline
Control	Gains by baking improvements into day-to-day process

Leverage on SHIP-HATS GitLab native DORA dashboards and / or CodeSCAPE to help **measure your baseline**. Remember to tag your environment deployment tier or name. The above framework can apply to other DORA metrices and is tool agnostic.

#2: DORA metrics x DevEx: Correlated!

High or Elite levels of DORA indicates:

- ✓ Good code quality
- ✓ Secure code
- ✓ High productivity
- ✓ Higher automation



Poor: Adds manual work
Good: Enables via automation
DORA metrics
Env
Practices
Tools
Dev Experience
</>
Good: Empowers via continuous improvement
Poor: hinders positive changes

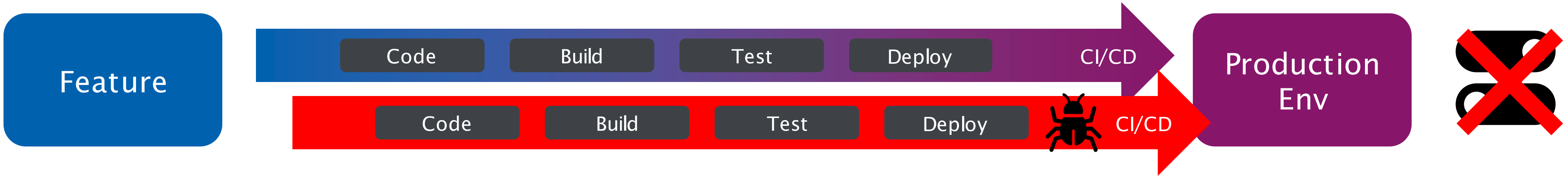
Good DevEx (DX) indicates:

- ✓ Better flow state
- ✓ Less cognitive load
- ✓ Constructive feedback loops
- ✓ Higher satisfaction

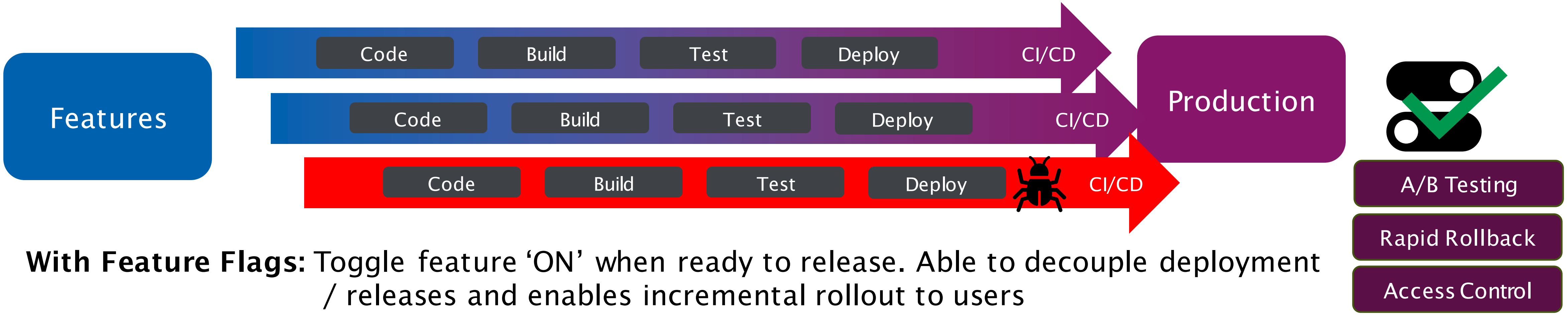
Consider both sides of the equation when designing improvements!

shiphats

#3: Implement Feature Flags



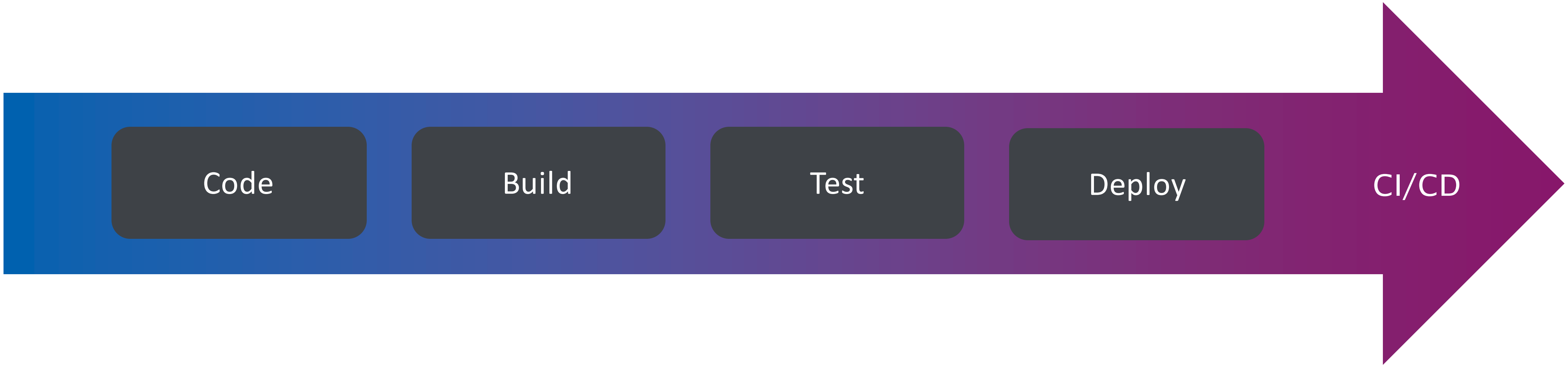
No feature flags: Results in rollback or emergency fixes. Blocks deployment in parallel



With Feature Flags: Toggle feature 'ON' when ready to release. Able to decouple deployment / releases and enables incremental rollout to users

GOVTECH SINGAPORE

#4: Allocate time for reviewing and improving

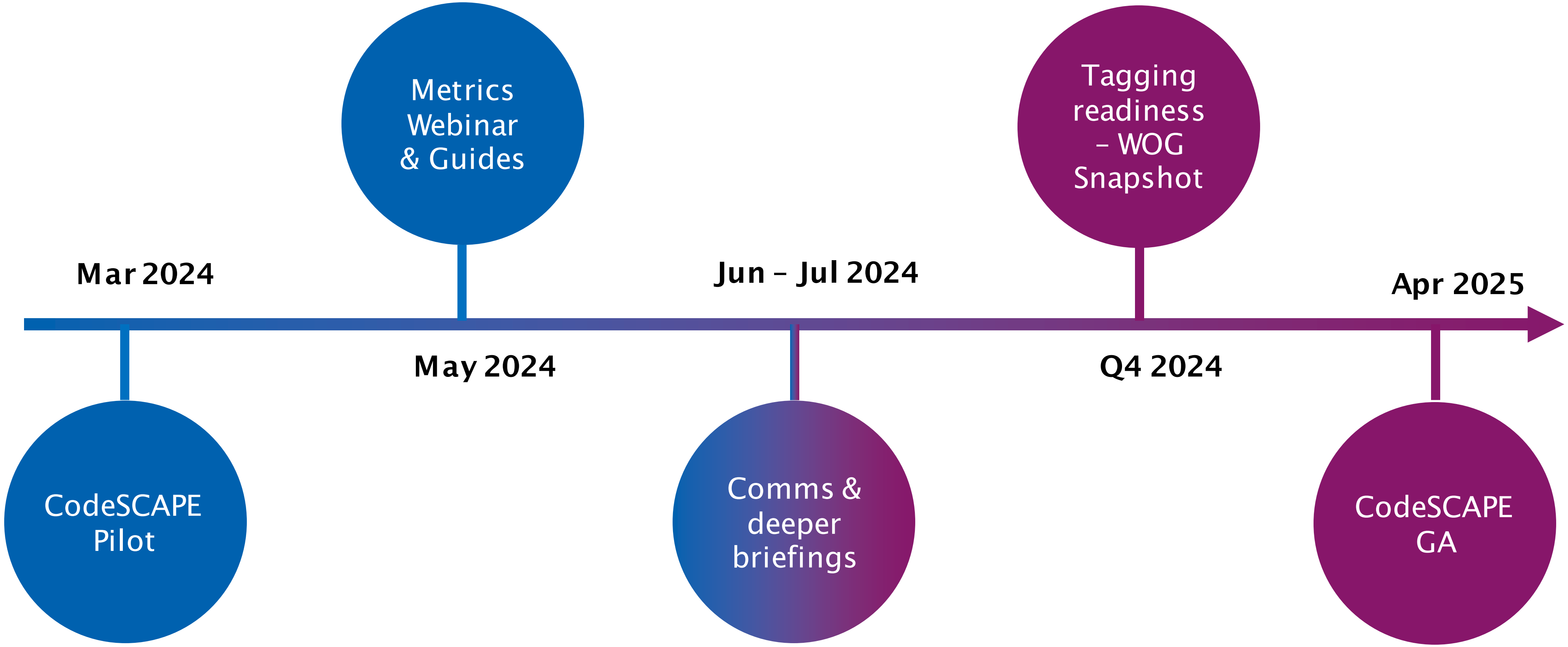


1. Plan time to review your pipeline
2. For each stage, what can be improved?
3. Are there security tests (SAST, DAST, SCA …) that can be incorporated to run automatically each time?
4. Are there tests that can be automated?

What is next?

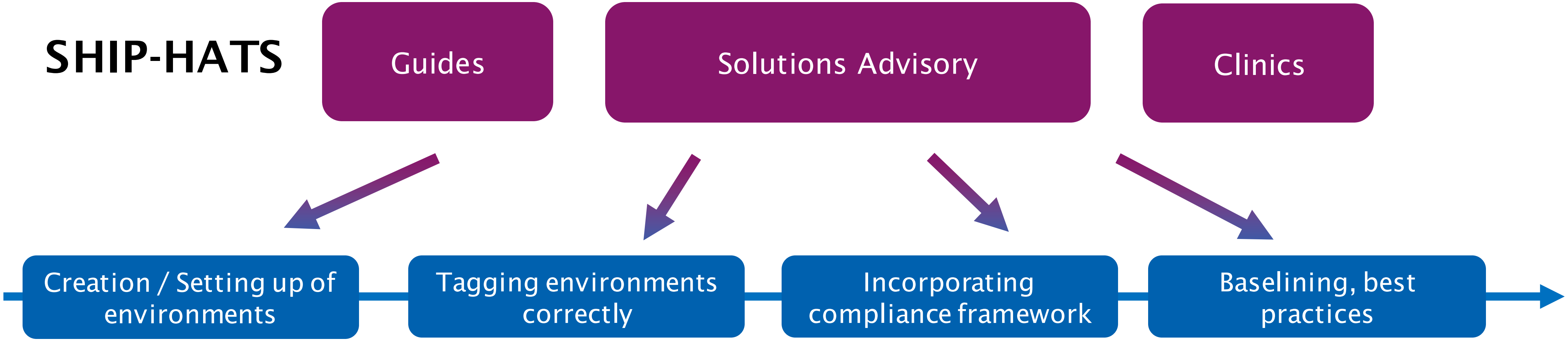
Leon Leow
Hudson Lee

Our Plans for baselining deployment frequency across WOG



The above is only valid for SHIP-HATS users

We can help you to improve your time-to-market



SHIP-HATS

Guides

Solutions Advisory

Clinics

Creation / Setting up of environments

Tagging environments correctly

Incorporating compliance framework

Baselining, best practices

Measure and Baseline with CodeSCAPE on DORA and DevSecOps maturity

Implemented correctly, usage of DORA metrics and measuring deployment frequency can help product teams shorten time from feature development to launch by delivering value to users quicker in smaller working increments and adapt quickly to changes.

3 Dimensions in DevOps Maturity Improvements

Process

- Value Stream Mapping → Critical Path & Bottleneck Removal
- Small Batches in each Release
- Reviews & Continuous Improvements (DMAIC approach)
- Objective driven → Remove complexities

People / Culture

- Shifting Left
- Communications & Blameless culture
- Supports from Management
- Well-defined Role Responsibly & Expectation
- Inner-sourcing & Reusability

Technologies

- Automate & Automate
- Collaboration & Communication
- AI-assisted Tools
- Observability Dashboards

Recap & Call to Action

- Get started on **tagging** your environments in SHIP-HATS GitLab in a proper manner
- Consider **using SHIP-HATS GitLab** for deployment to your Prod Environment
- **Access resources** provided and materials to learn.
- **Reach out** to us (enquiries_ship@tech.gov.sg) if you need support from our Solutions Advisory team.
- **Attend further briefings** if you are a SHIP-HATS user.

Q & A

Scan the code and add questions!

https://pigeonhole.at/SHIPHATS

Share your feedback!

https://go.gov.sg/sgts-events-survey

https://go.gov.sg/sgts-events-survey

Thank You