

# **Fantastic Supply–Chain Vulnerabilites**

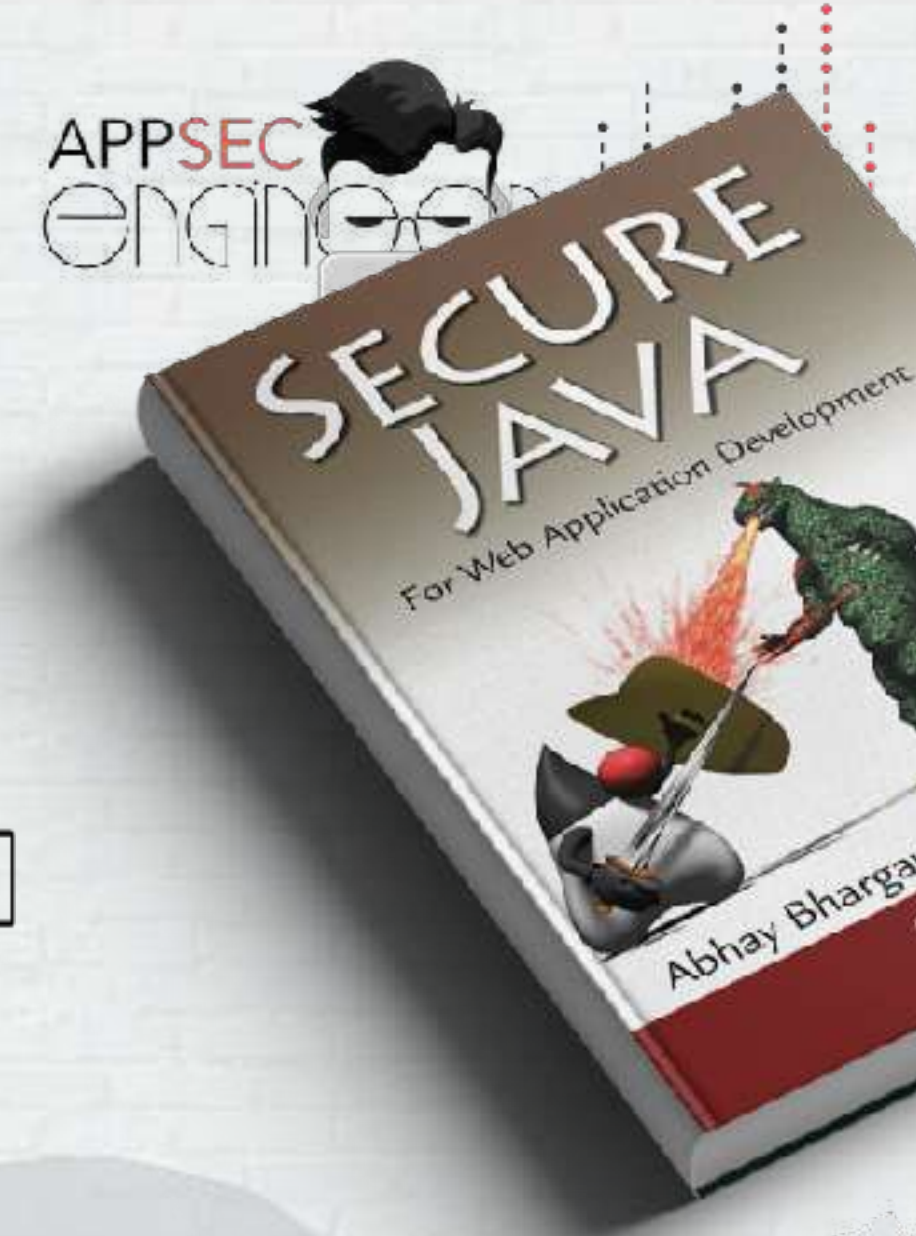
Abhay Bhargav



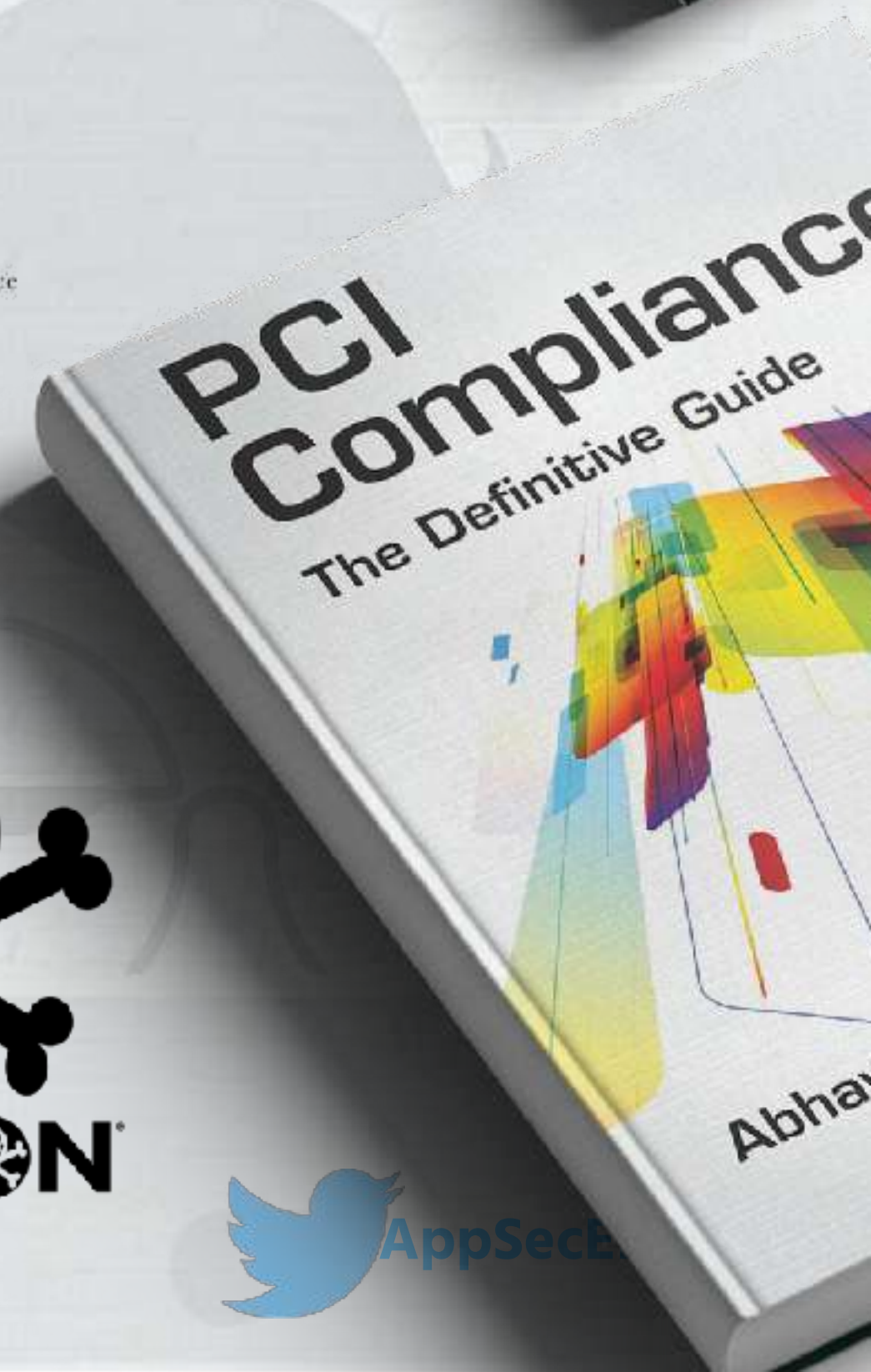


# Yours Truly

- Founder @ we45 and AppSecEngineer
- Chief Architect - Orchestron
- Avid Pythonista and AppSec Automation Junkie
- Trainer/Speaker at DEF CON, BlackHat, OWASP Events, etc world-wide
- Lead Trainer - we45 Training and Workshops
- Co-author of Secure Java For Web Application Development
- Author of PCI Compliance: A Definitive Guide



DVFaaS  
Damn Vulnerable Functions as a Service





# Community Initiatives



📢 Youtube Channel: [youtube.com/appsecengineer](https://youtube.com/appsecengineer)

📖 Blog: [we45.com/blog](https://we45.com/blog)

💡 Talks/Workshops at several Events





# My talk...





# My talk...





“the network of all the individuals, organizations, resources, activities and technology involved in the creation and sale of a product.”

–Definition of Supply Chain



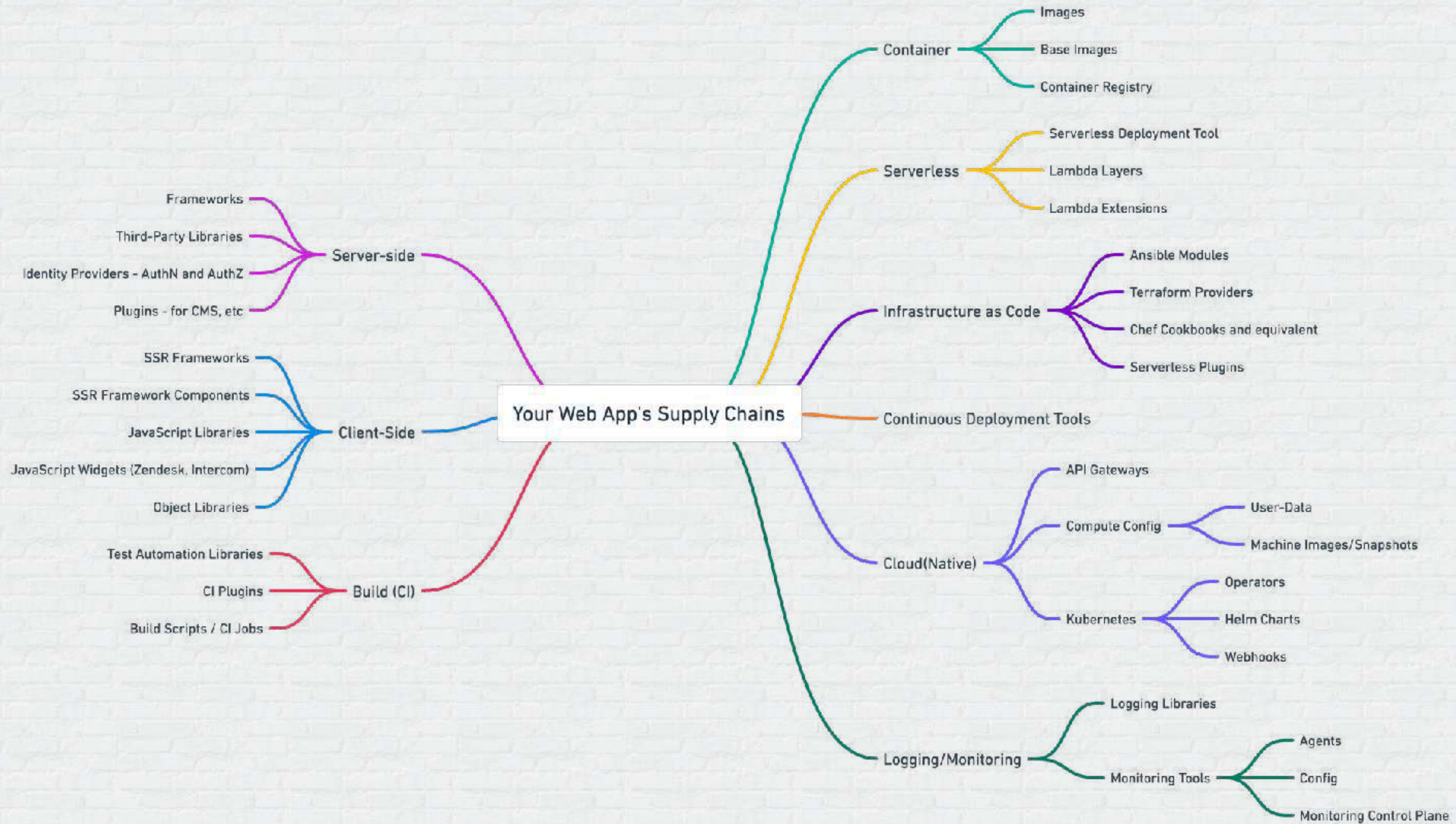


**“A software supply chain is composed of the components, libraries, tools, and processes used to develop, build, and publish a software artifact.”**

–Usenix Paper coauthored by Dan Geer



# Your Application's Supply-Chains






# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks




Trending Innovation Security Business Finance Education Home & Office More

PART OF A ZDNET SPECIAL FEATURE: **SECURITY AND PRIVACY: NEW CHALLENGES**

[Home](#) | [Innovation](#) | [Security](#)

## Anatomy of the Target data breach: Missed opportunities and lessons learned

Target hasn't publicly released all the details of its 2013 data breach, but enough information exists to piece together what likely happened and understand how the company could have prevented the hack.


 Written by Michael Kassner on Feb. 2, 2015



# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks









Trending Innovation Security Business Finance Education Home & Office More

Home | Innovation | Security


## Wi-Fi hack caused TK Maxx security breach

The biggest loss of credit-card data in history was brought about largely because of lax wireless LAN security, it has emerged


 Written by Tom Espiner, Senior Reporter on May 8, 2007


MUST READ



Hackers who stole 45 million customer records from the parent company of TK Maxx did so by breaking into the retail company's wireless LAN , it emerged on Monday.

 abhaybhargav

Copyright © AppSecEngineer 2022

 AppSecEngineer



# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks

[Home](#) > [Security](#) > [Ransomware](#)

ANALYSIS

## What is WannaCry ransomware, how does it infect, and who was responsible?

Stolen government hacking tools, unpatched Windows systems, and shadowy North Korean operatives made WannaCry a perfect ransomware storm.



By **Josh Fruhlinger**

Contributing writer, CSO | 30 AUGUST 2018 19:22 IST





# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks

## NotPetya: How a Russian malware created the world's worst cyberattack ever

NotPetya malware spread like wildfire across the world, eating into every electronic equipment, computers, extracting data and demanding exorbitant amounts for recovery in form of Bitcoins

### Topics

Notpetya Ransomware Attack | Cybersecurity | Hackers

Aparna Banerjea | New Delhi  
Last Updated at August 27, 2018 12:30 IST

Follow us on



Market

LATEST NEWS

IN THIS SECTION

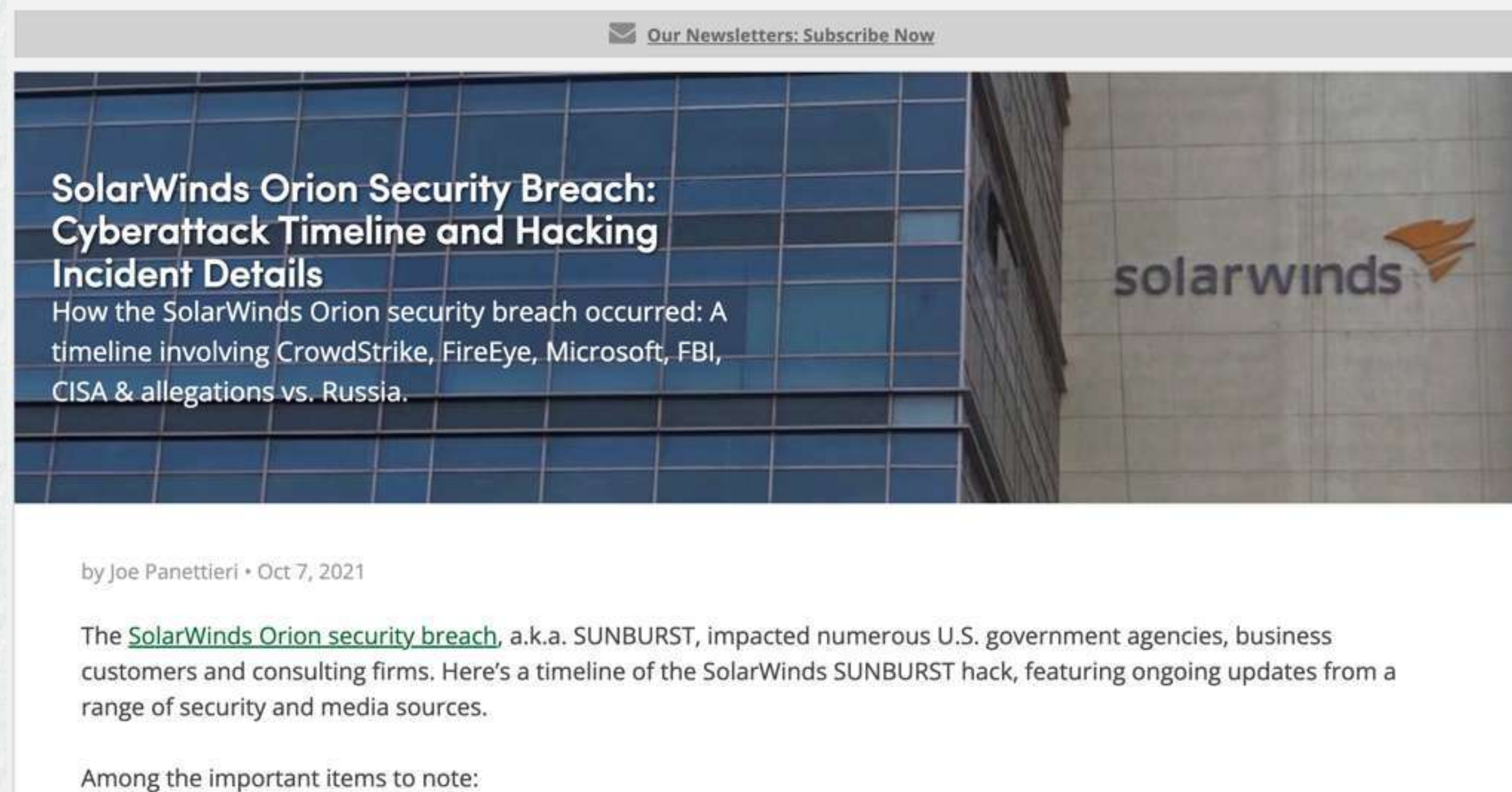




# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks





# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks





# A Recent History of Supply-Chain Attacks



# A Recent History of Supply-Chain Attacks

 **REUTERS®**

World ▾ Business ▾ Legal ▾ Markets ▾ Breakingviews Technology ▾ Investigations More ▾

April 20, 2021  
5:21 AM GMT+5:30  
Last Updated a year ago

Technology

## Codecov hackers breached hundreds of restricted customer sites - sources

By Joseph Menn and Raphael Satter

4 minute read

Register now for FREE unlimited

SAN FRANCISCO, April 19 (Reuters) - Hackers who tampered with a software development tool from a company called Codecov used that program to gain



# A Recent History of Supply-Chain Attacks



# Today's Agenda





# Today's Agenda

- Three Stories



# Today's Agenda

- Three Stories
- From different phases of the SDLC



# Today's Agenda

- Three Stories
- From different phases of the SDLC
- With completely different supply-chain implications



# Pre-Commit Supply-Chain Attacks

## Pre-Commit Supply-Chain Attacks

Malicious Dependency - Local Command Exec

Dependency Confusion

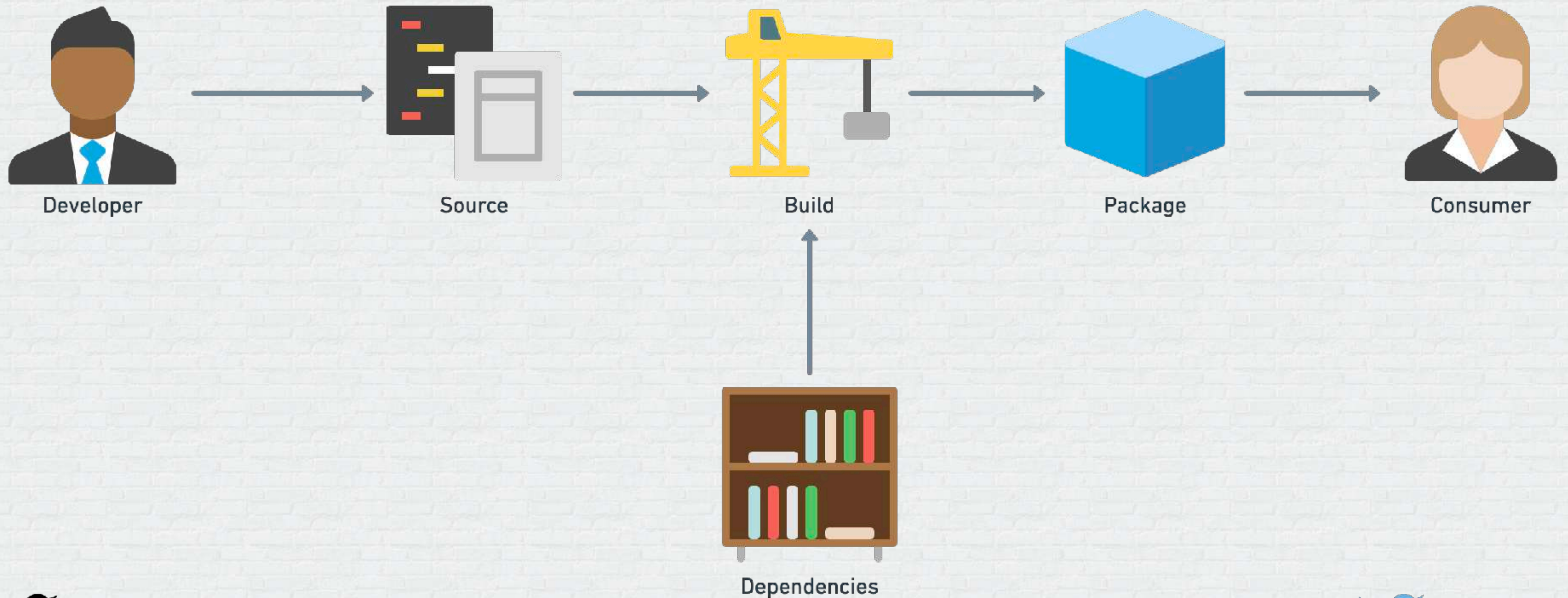
Malicious Git Hooks

Malicious/Compromised Infrastructure-as-Code  
Manifests

Compromised IDE Plugins/Dev-tooling



# Supply-Chain Lifecycle





# Supply-Chain Lifecycle

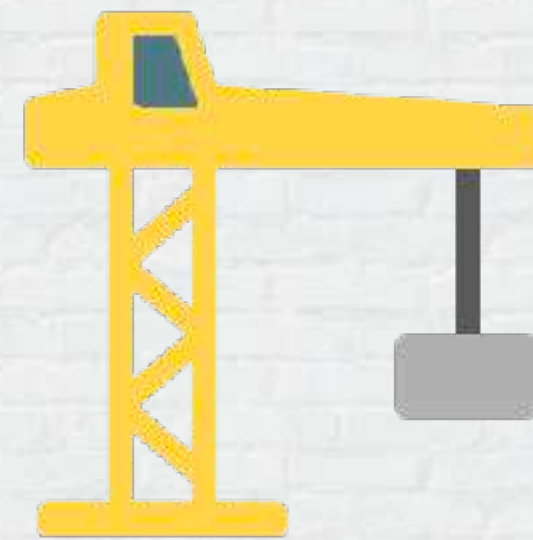
Dependency Confusion  
Malicious Git Hooks  
Malicious Terraform Modules



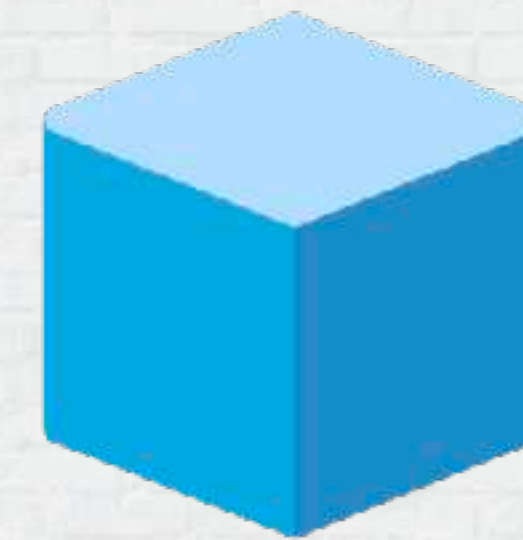
Developer



Source



Build



Package



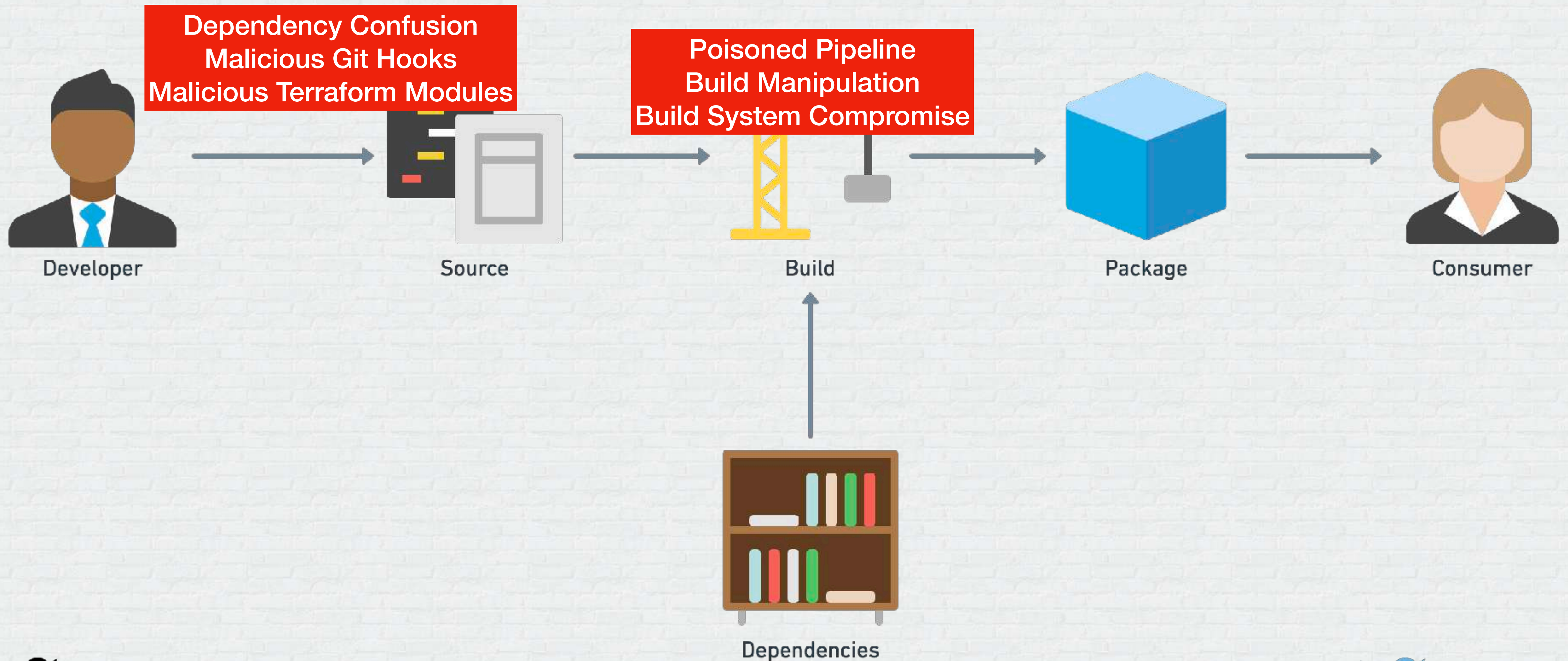
Consumer



Dependencies

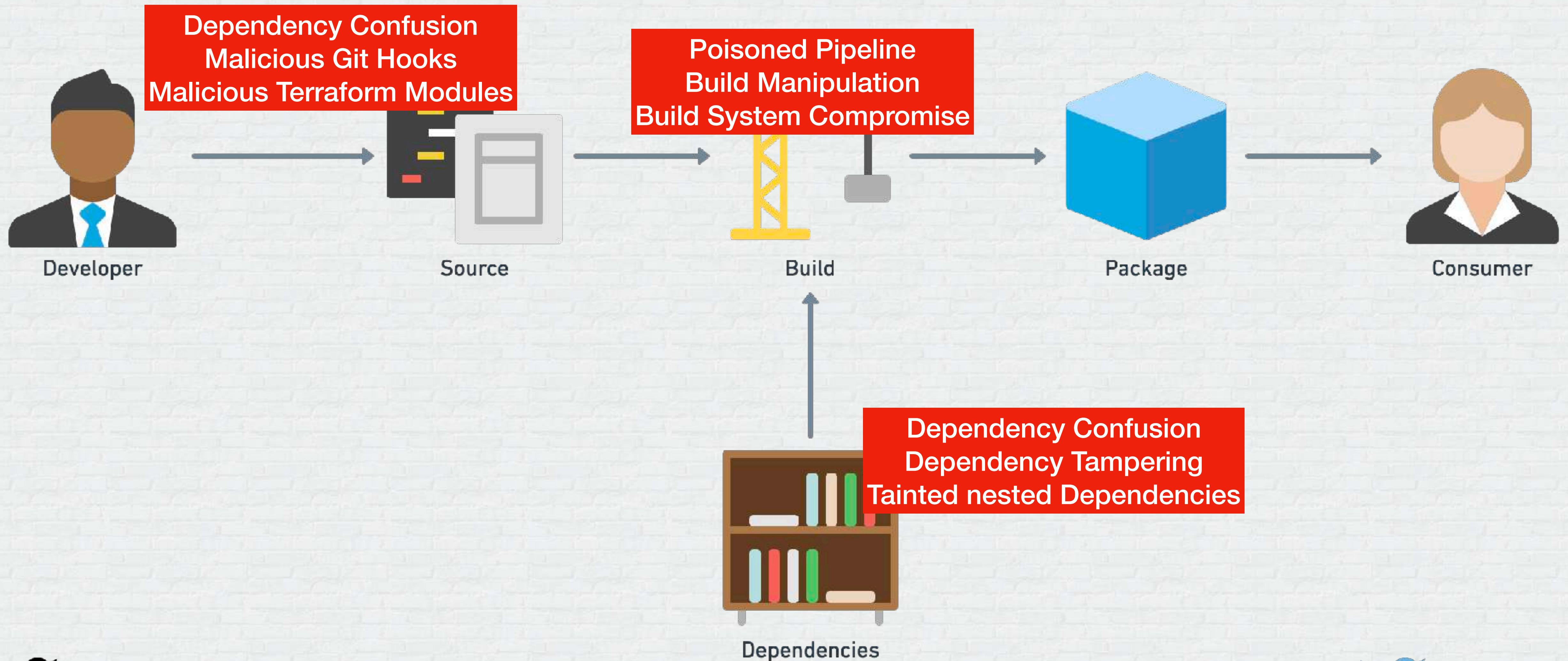


# Supply-Chain Lifecycle



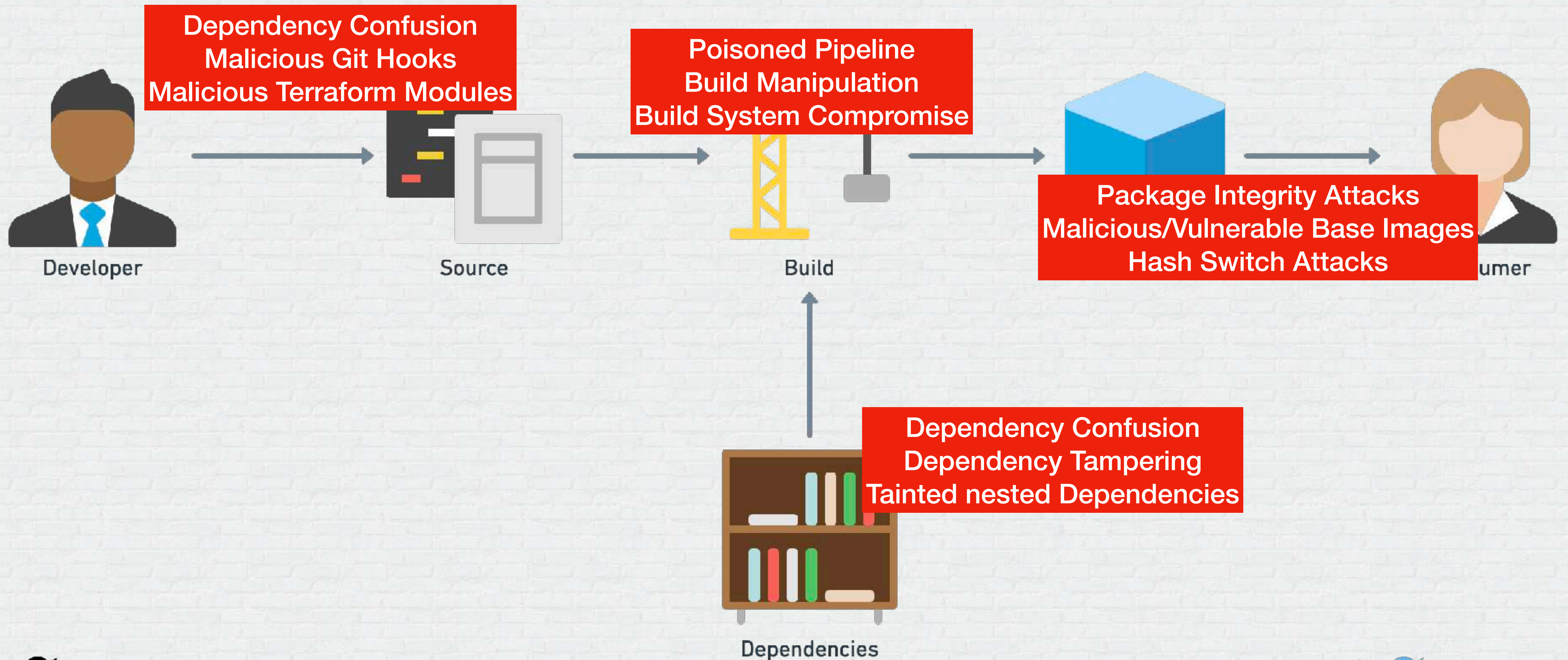


# Supply-Chain Lifecycle





# Supply-Chain Lifecycle





# Dependency Confusion





# How does it work?



# How does it work?



Developer



# How does it work?



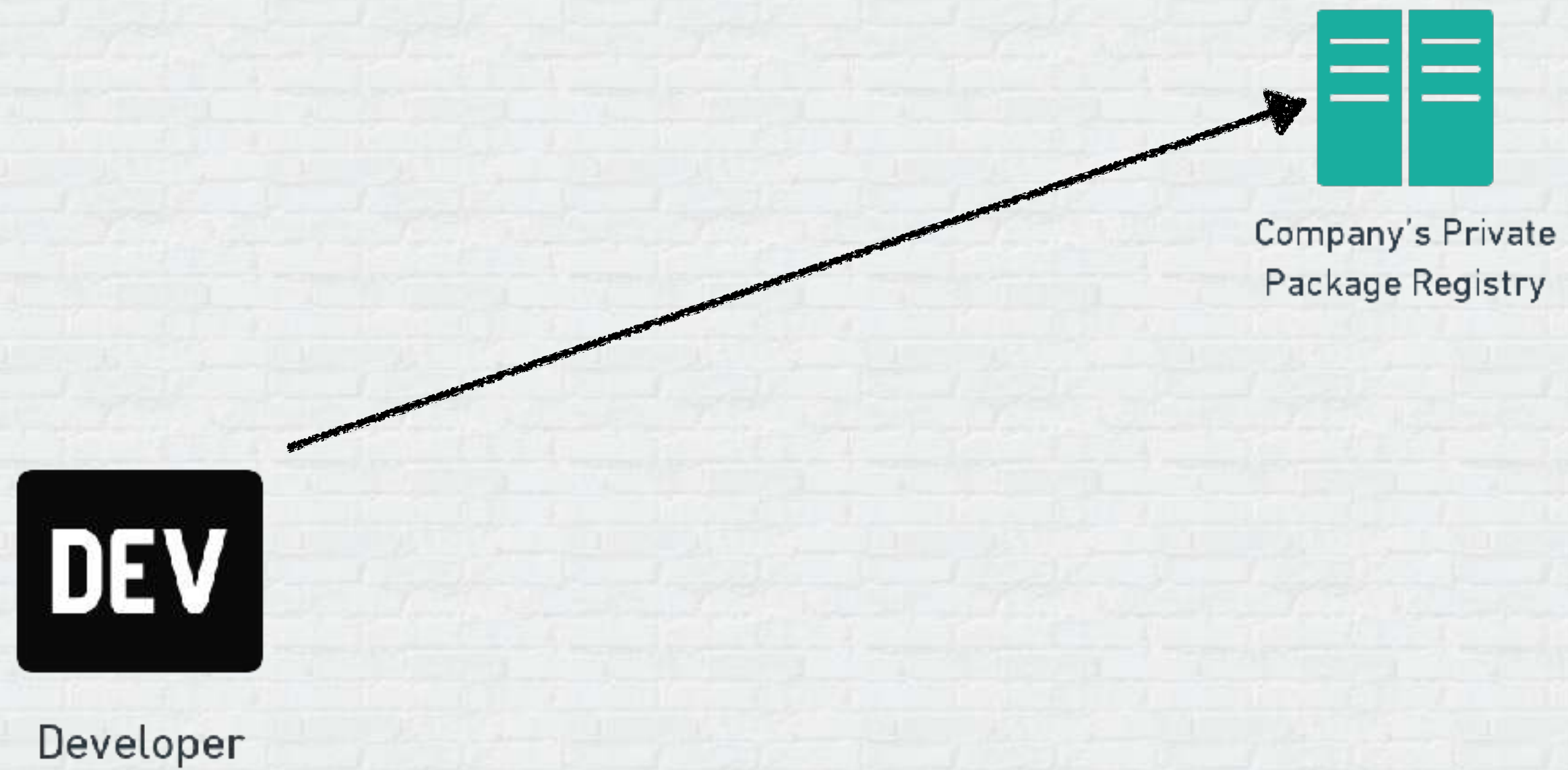
Company's Private  
Package Registry



Developer

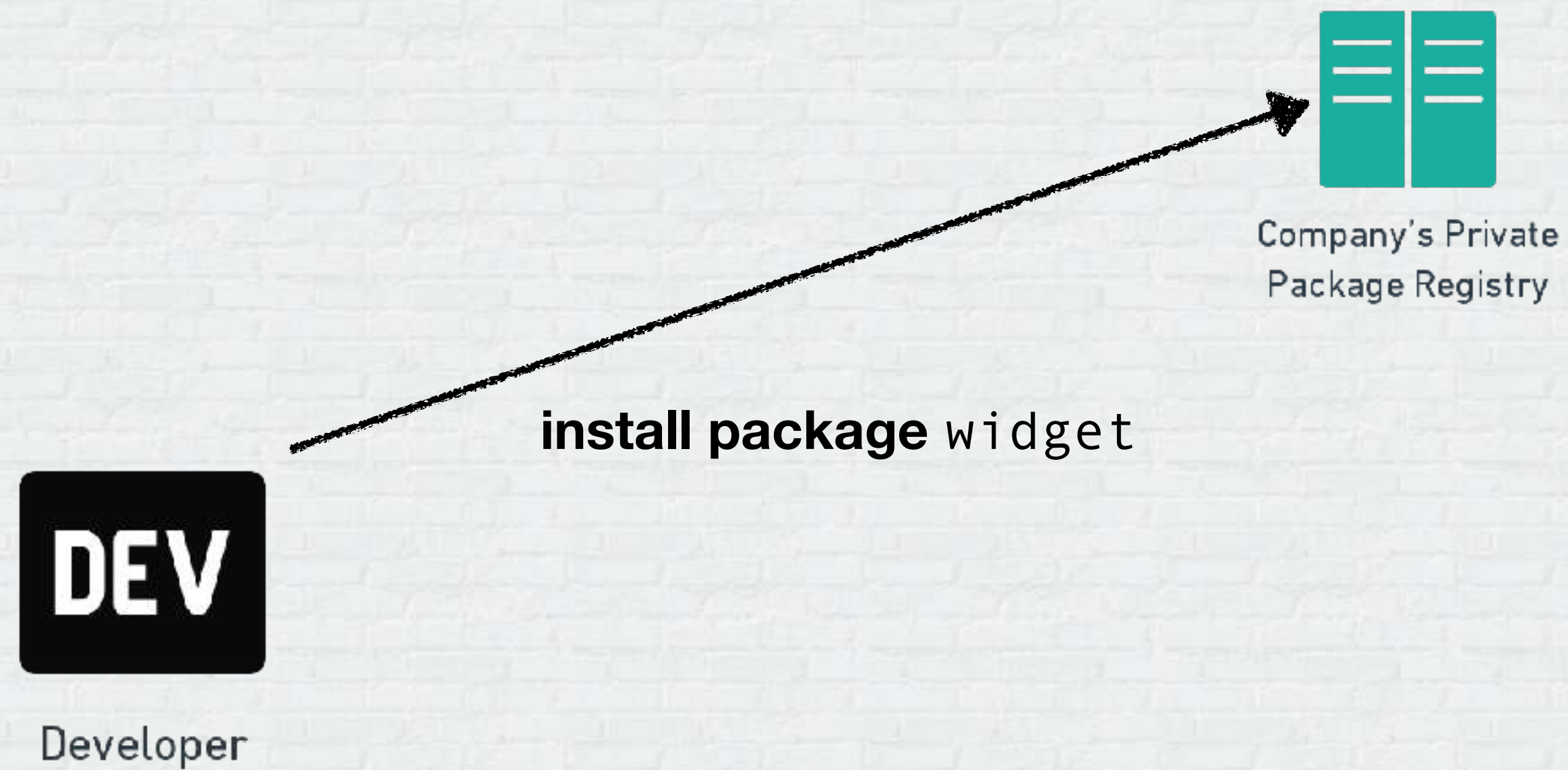


# How does it work?



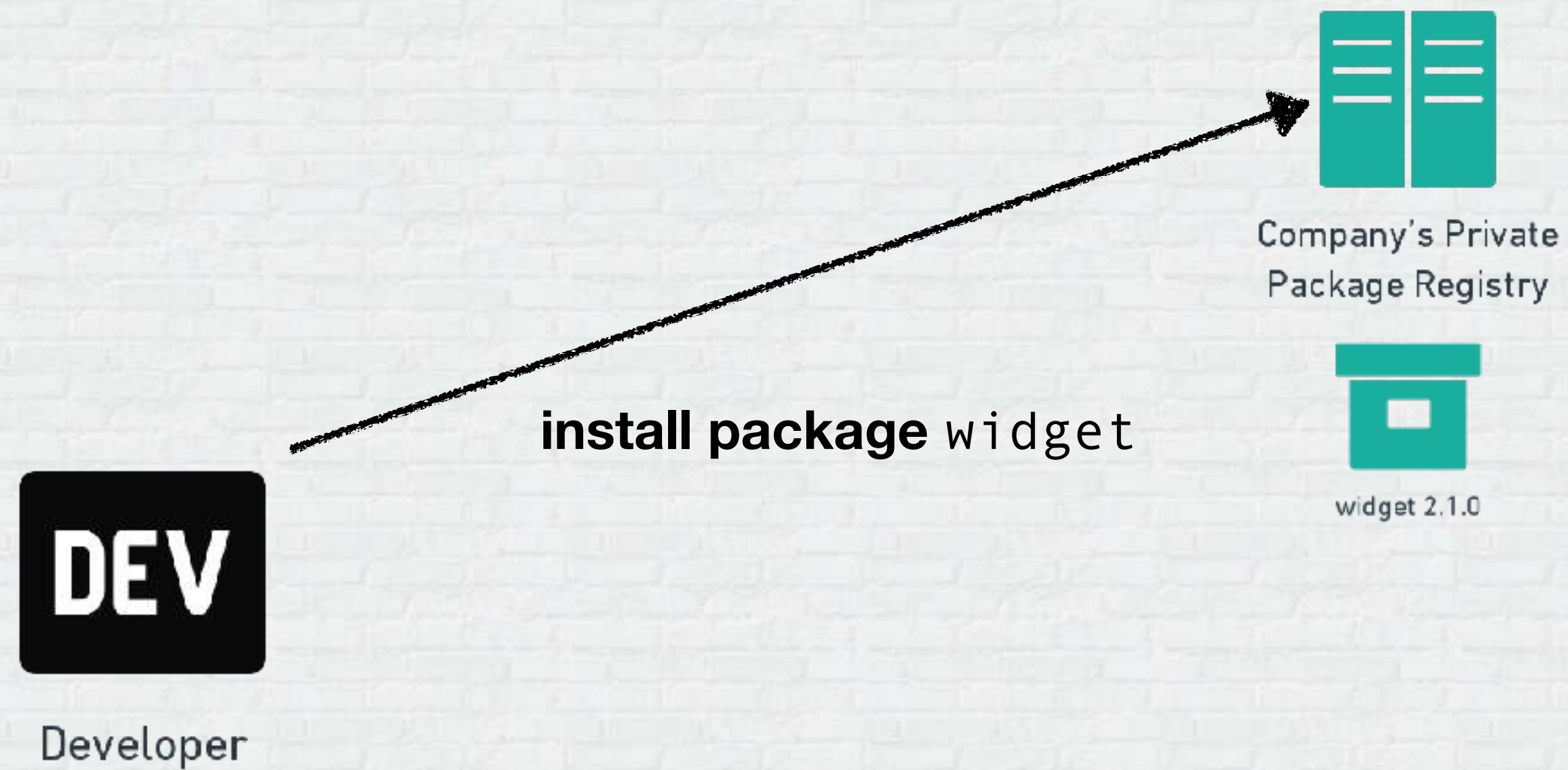


# How does it work?



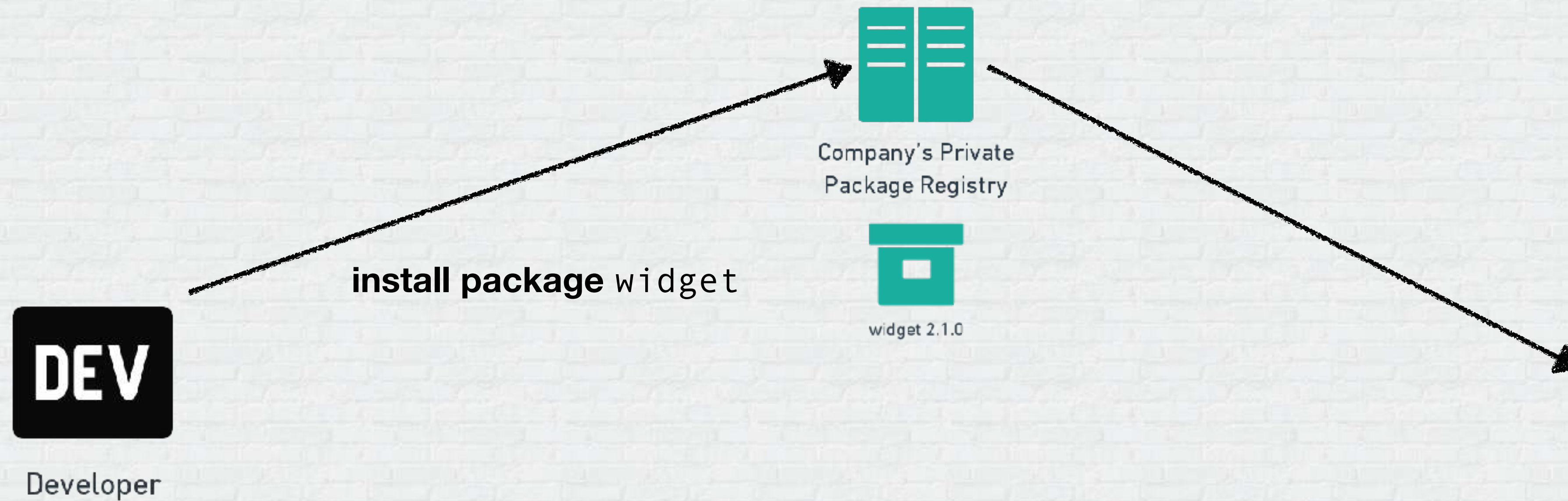


# How does it work?



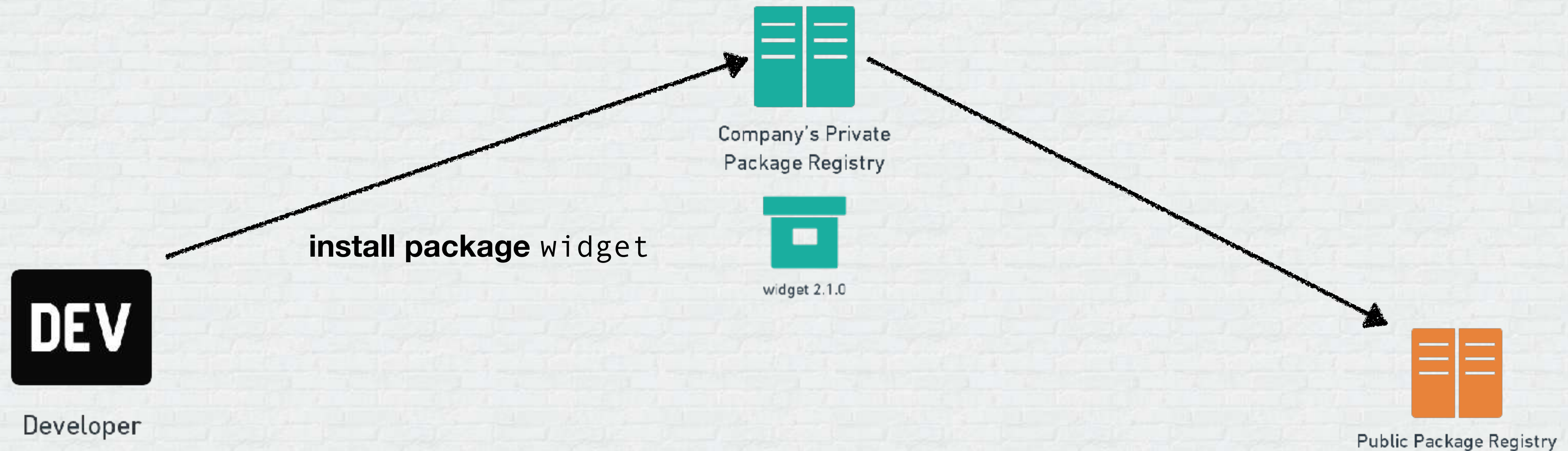


# How does it work?



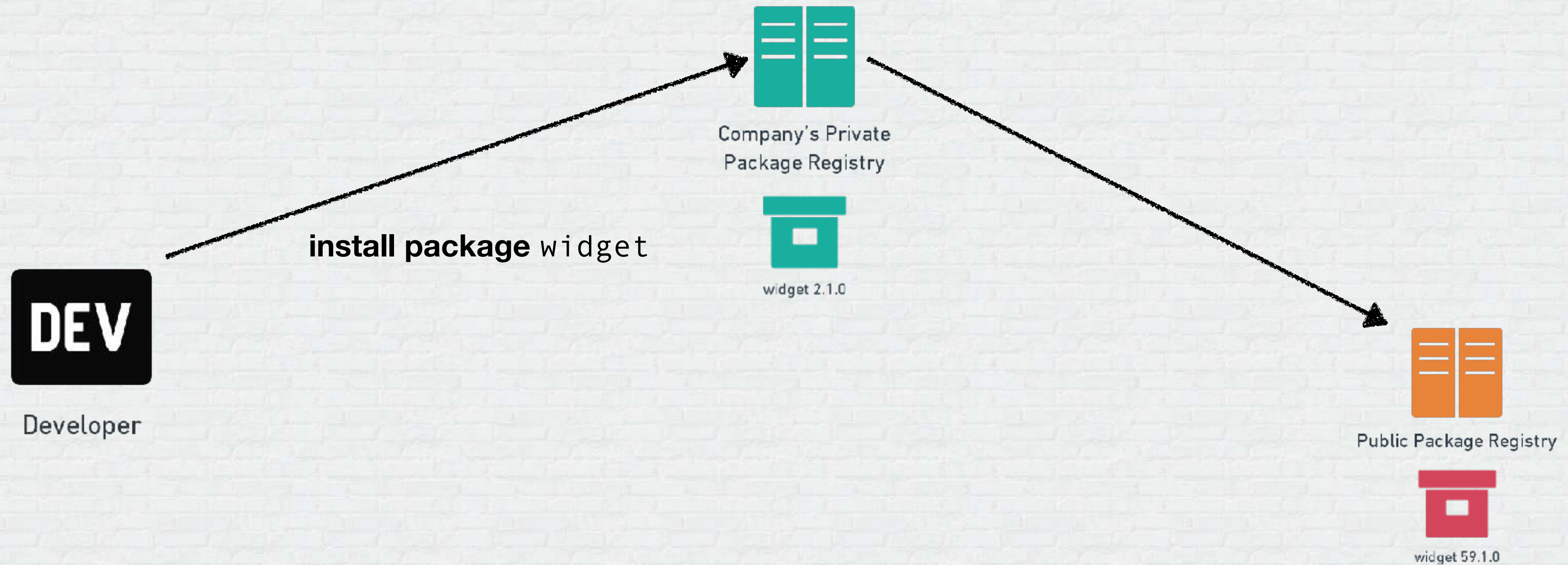


# How does it work?



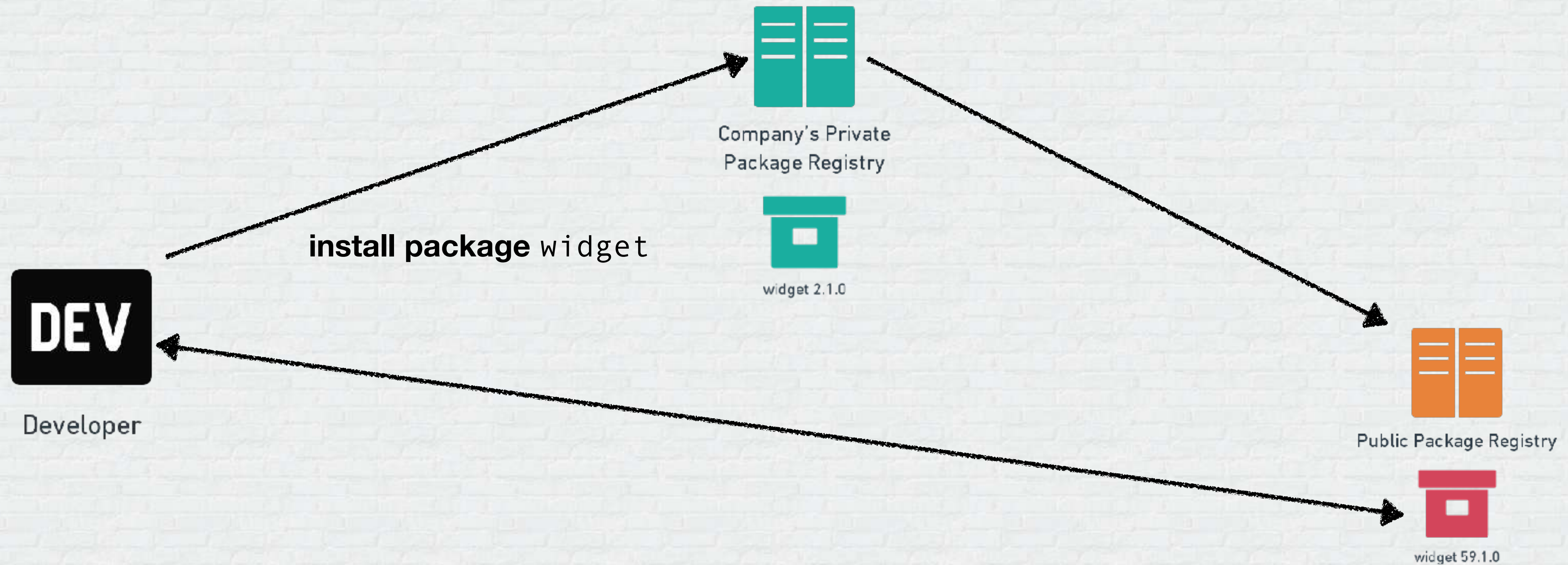


# How does it work?



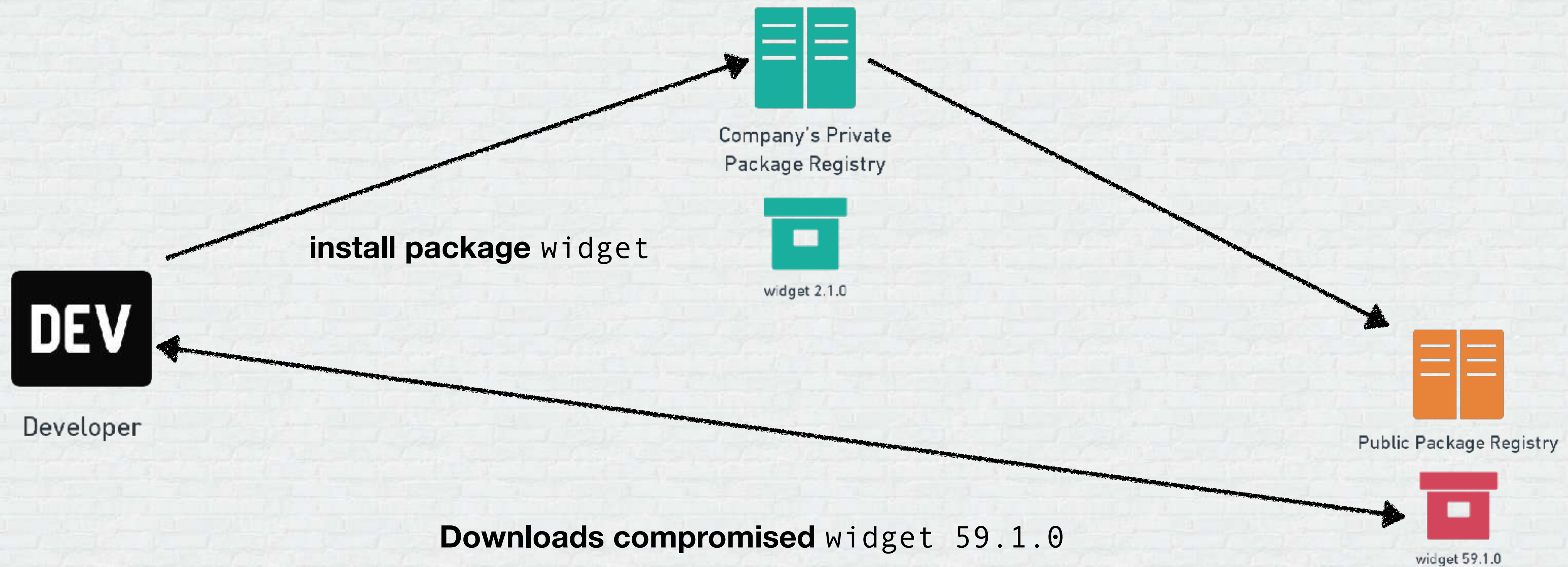


# How does it work?





# How does it work?





# Who does it affect?

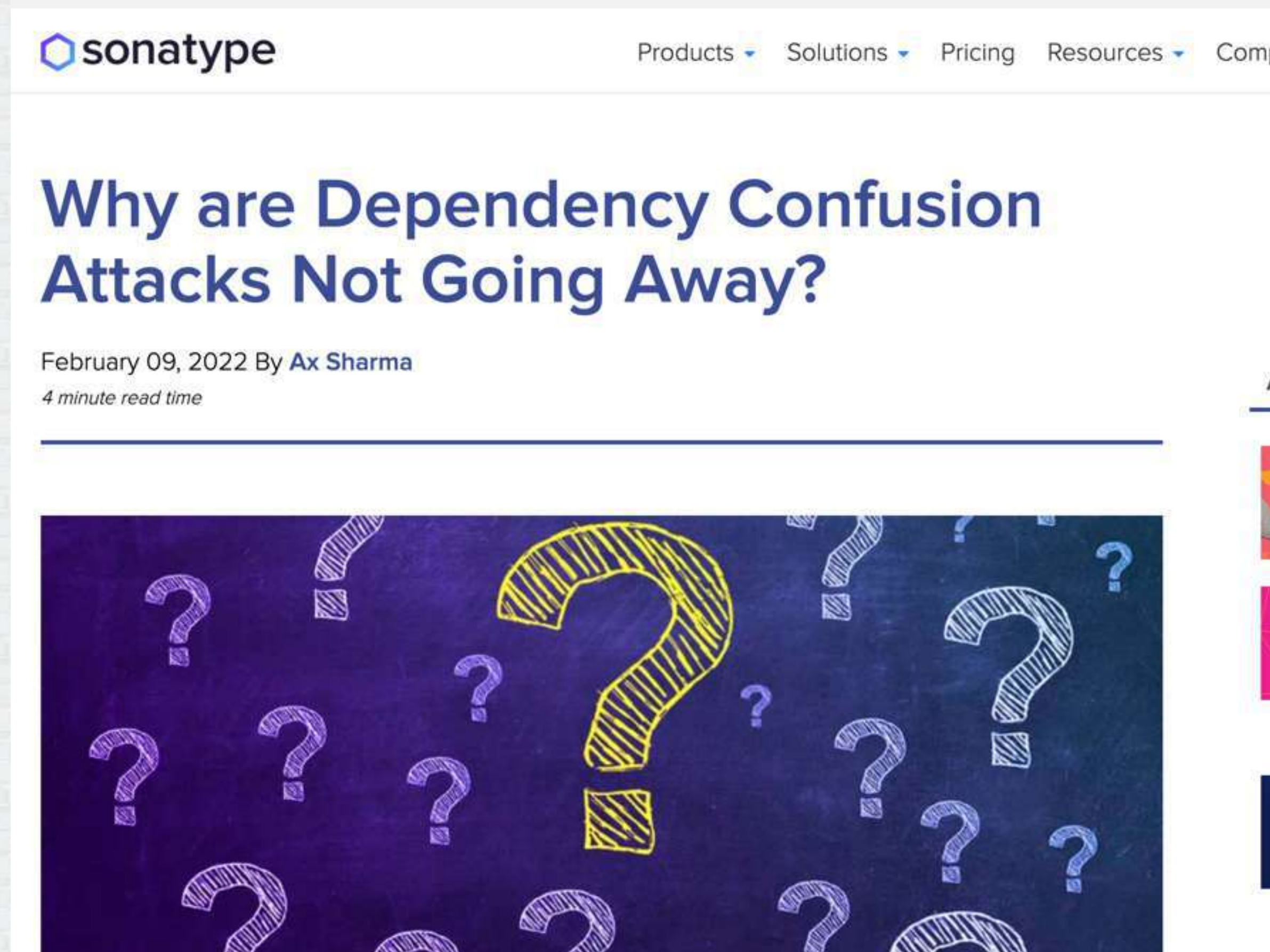
- Orgs with private packages in private repositories
- Orgs with private packages in artifactories (JFrog, etc)



# Examples



# Examples





# Examples



# Examples

REVERSINGLABS BLOG

Threat Research | May 10, 2022

## Update: NPM dependency confusion hacks target German firms

Research by ReversingLabs suggests that dependency confusion attacks on npm repositories have been used to compromise German firms - exposing an apparent red team exercise.



BLOG AUTHOR  
Paul Roberts,  
Cyber Content Lead at ReversingLabs. [Read More...](#)








# Examples



# Examples





# Examples



# Lab: Dependency Confusion





# Terror with Terraform





# Terraform Terminology

- Providers => Plugins to interact with cloud environments. Found in the Terraform Registry (example: AWS)
- Modules => Container for multiple resources that are used together (example - your app stack with specific resources, network and variable definitions)
- Resources => API Resources that refer to resources in specific cloud providers (example `aws_ssm_parameter`)



# Provider Types

- Community Providers - Anyone can submit. Will be signed. No additional verification
- Verified Providers -> Verified by Hashicorp Alliances Team
- Official Provider -> Managed by Hashicorp



# Terraform Modules

- Can be loaded from local directories
- Can be loaded from registry
- Can be loaded from Git repos
- No concept of verified or unverified Modules
- No signature for Terraform modules



# Implant Mechanisms



# Implant Mechanisms

- Developers using module in \$environment



# Implant Mechanisms

- Developers using module in \$environment
- Developers using providers that use the module



# Implant Mechanisms

- Developers using module in \$environment
- Developers using providers that use the module
- Terraform containers using module (prebuilt)



# Implant Mechanisms

- Developers using module in \$environment
- Developers using providers that use the module
- Terraform containers using module (prebuilt)
- Cross-Build Injection - Forced use of terraform module



# What about IaC SAST?





# What about IaC SAST?

- Bypassing IaC SAST rules entirely possible with base64-encoding, other techniques



# What about IaC SAST?

- Bypassing IaC SAST rules entirely possible with base64-encoding, other techniques
- Its all about studying the rules and identifying bypasses based on the checks



# Recommendations

- Only use modules/providers that have been audited
- Run SAST rules on modules to identify possible anomalies - use of base64, credential usage, etc.
- When running with CI/CD systems, try and build hermetically to avoid possible tainting of modules/providers
- Commit and Leverage lock files across the source artefact supply-chain



# Cluster Buster: Kubernetes Admission Control Scandal



# Typical Players – Containers and K8s



GitOps CD Tools



# Typical Players – Containers and K8s





# Typical Players – Containers and K8s



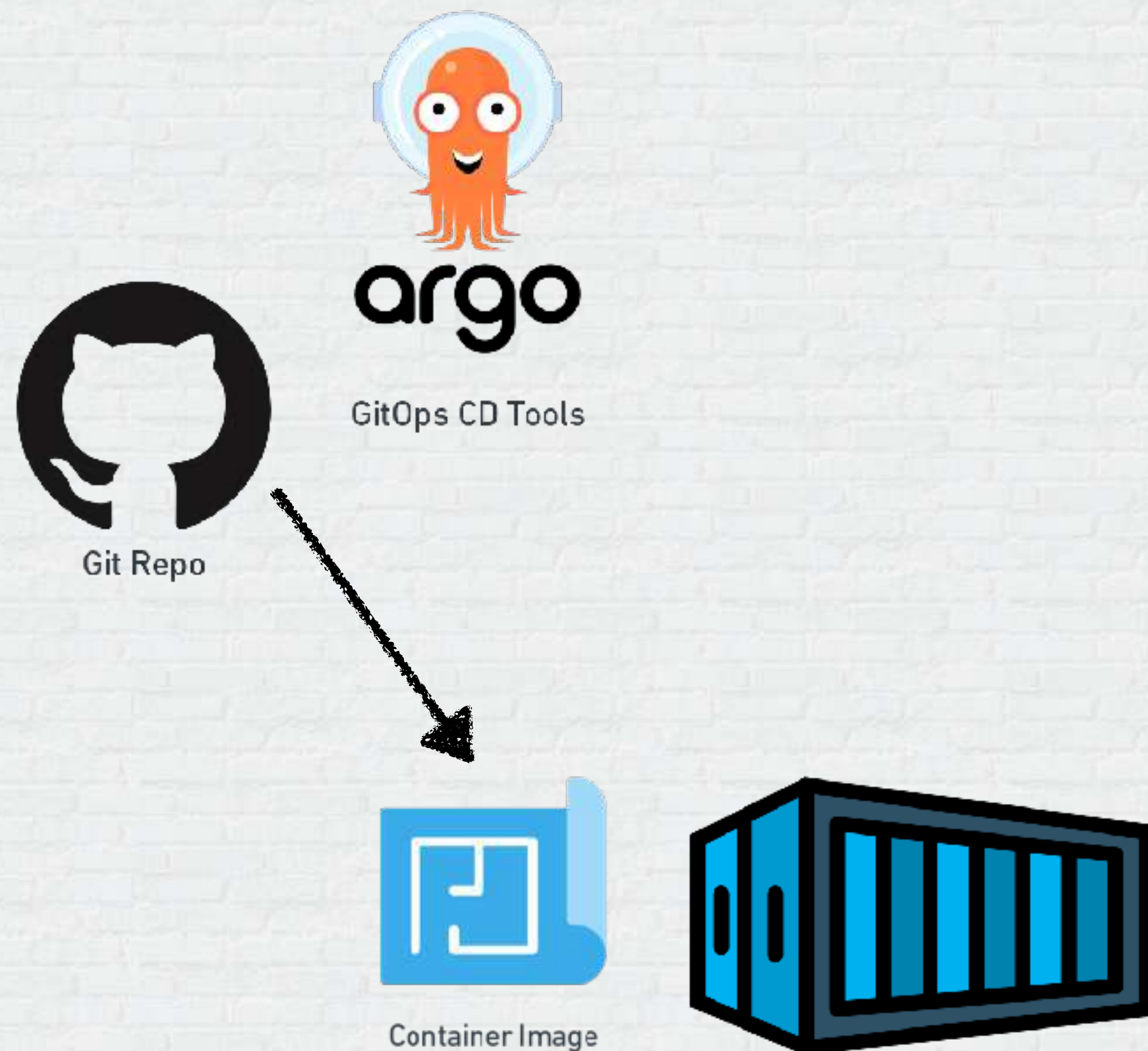


# Typical Players – Containers and K8s



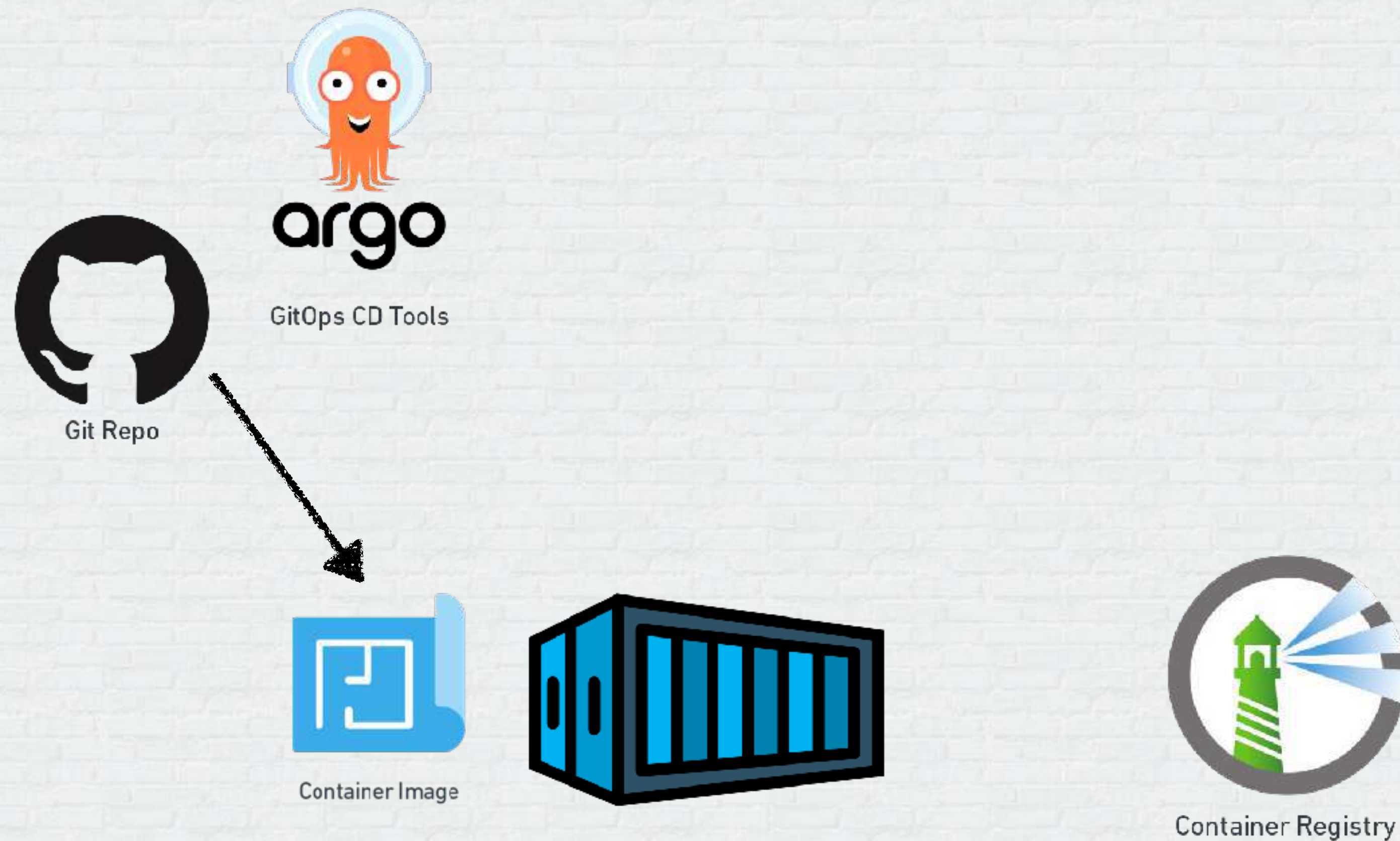


# Typical Players – Containers and K8s



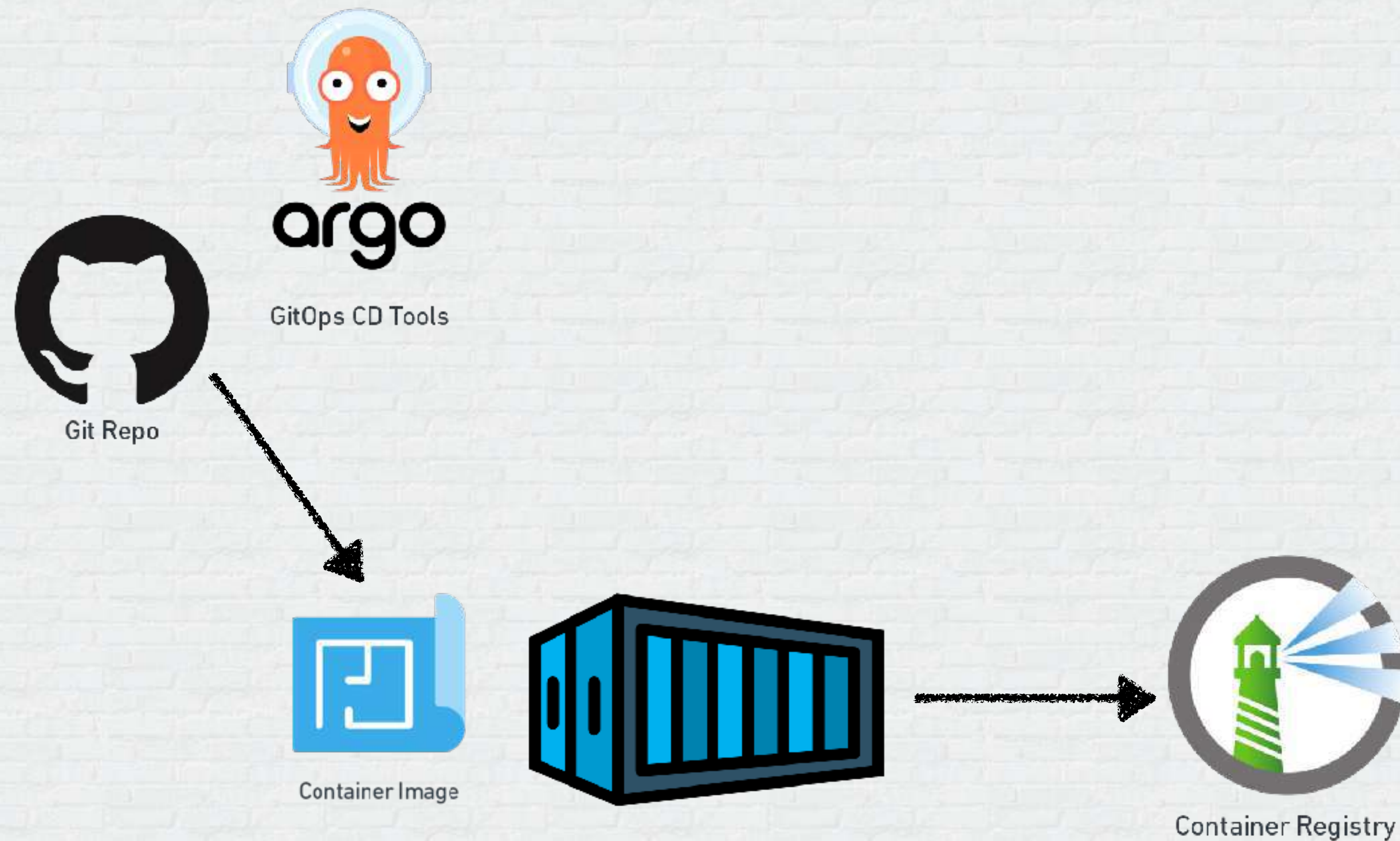


# Typical Players – Containers and K8s



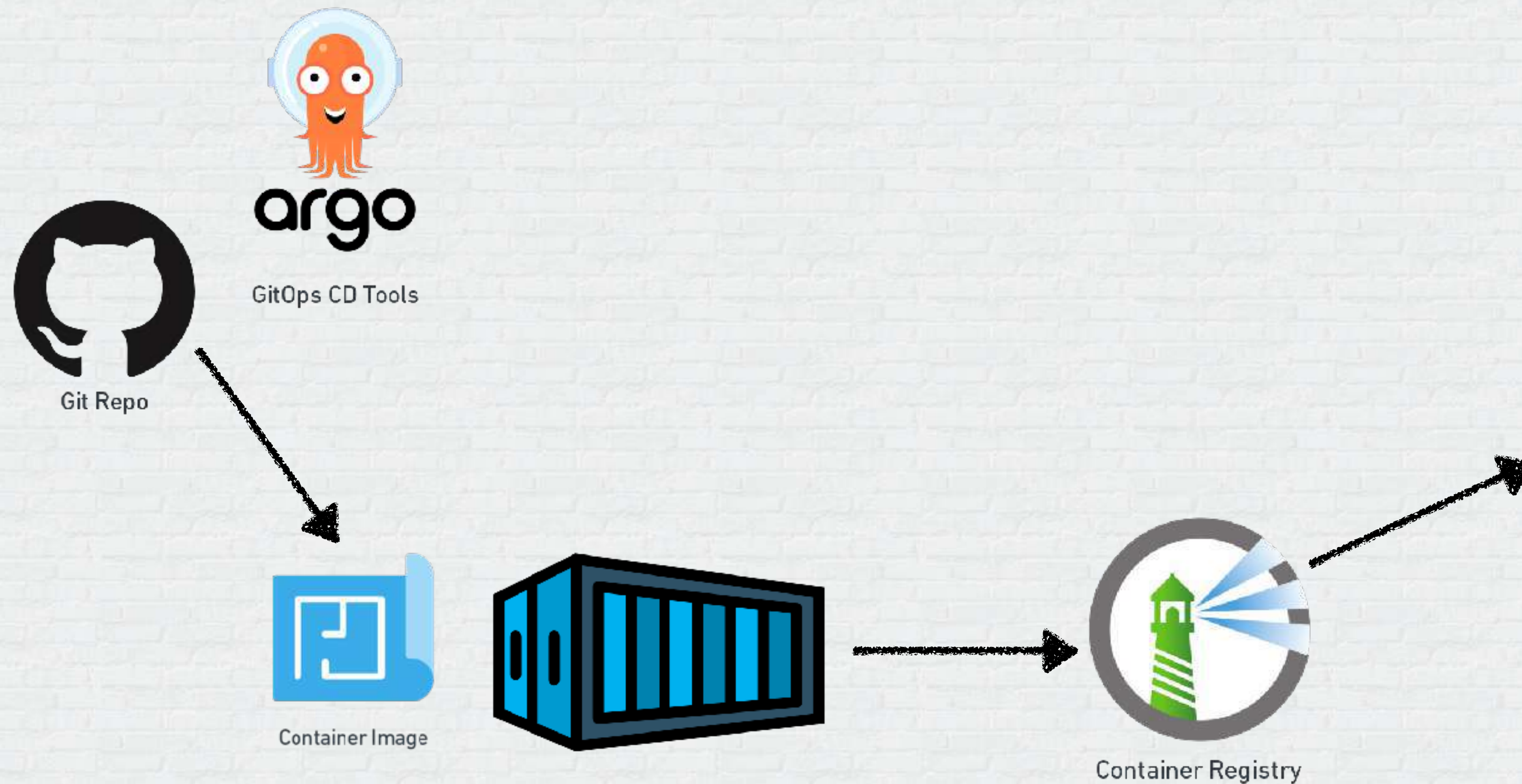


# Typical Players – Containers and K8s



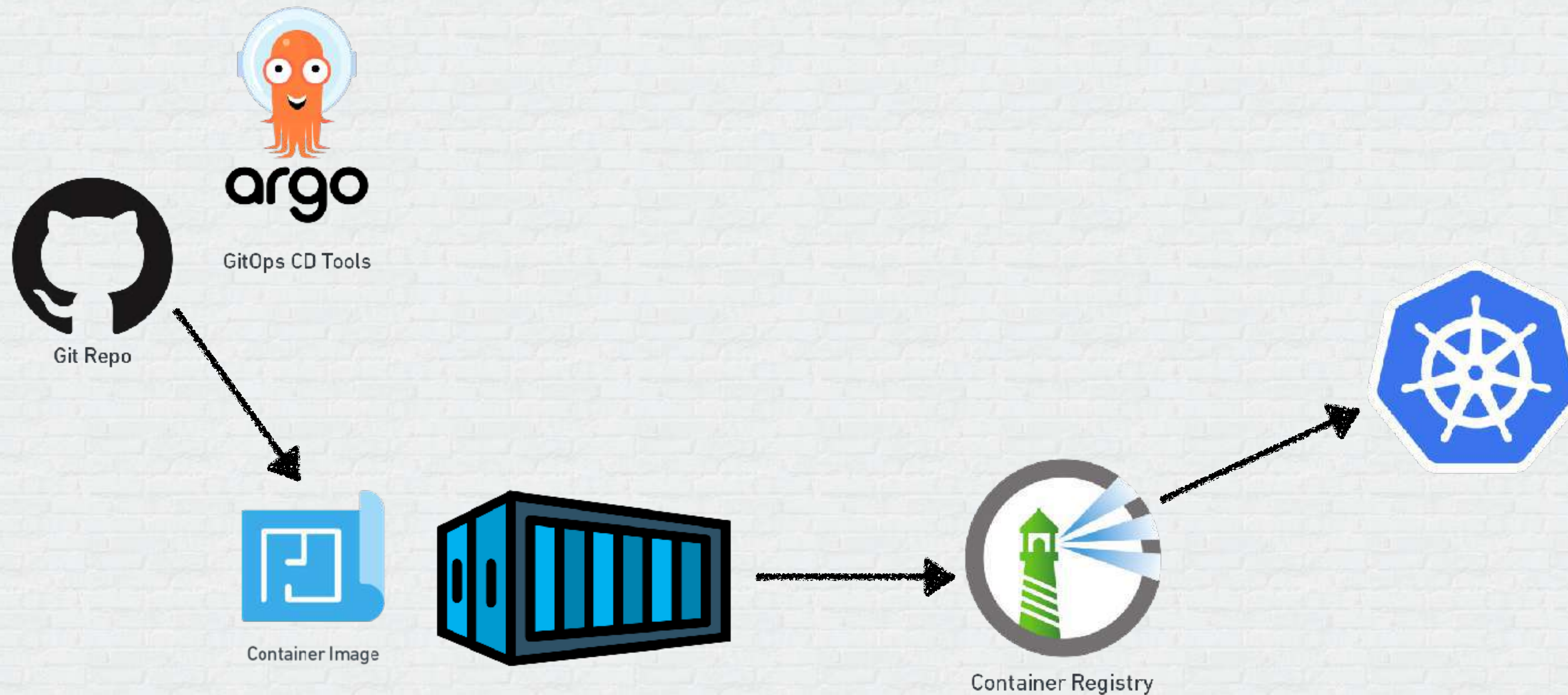


# Typical Players – Containers and K8s



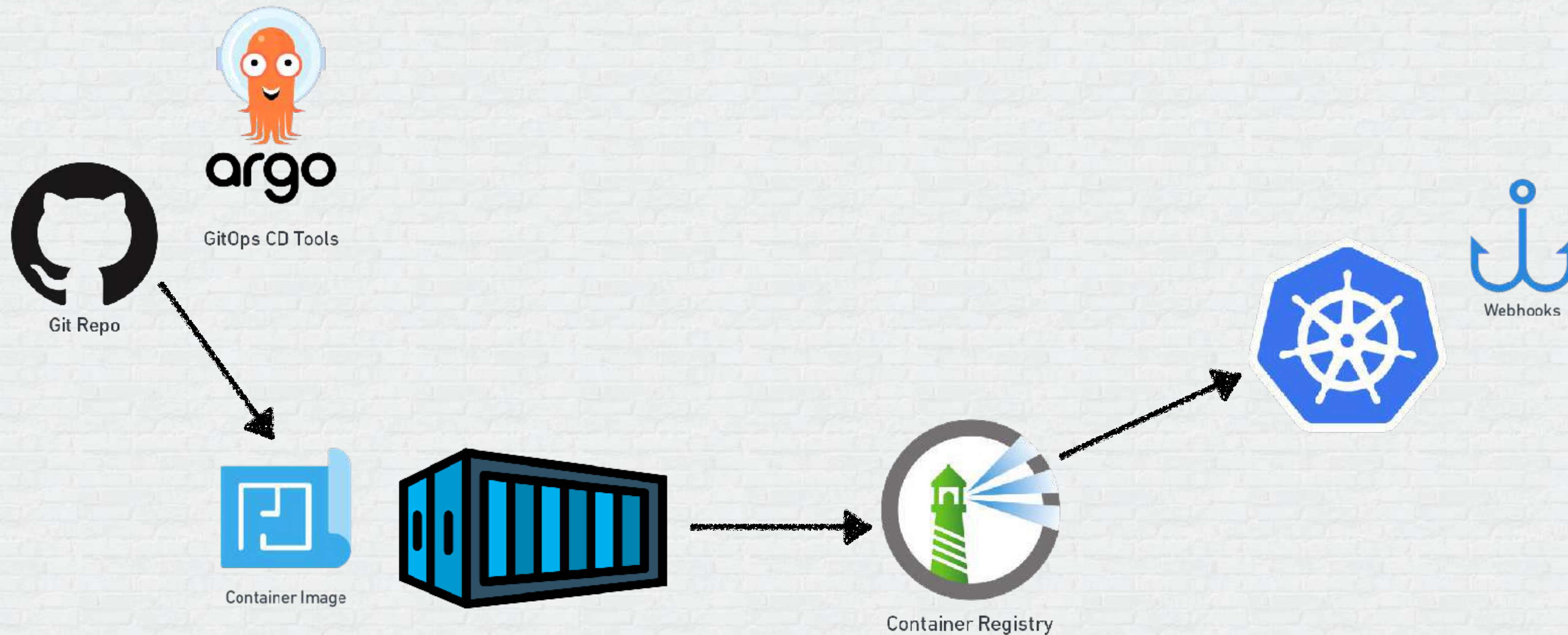


# Typical Players – Containers and K8s



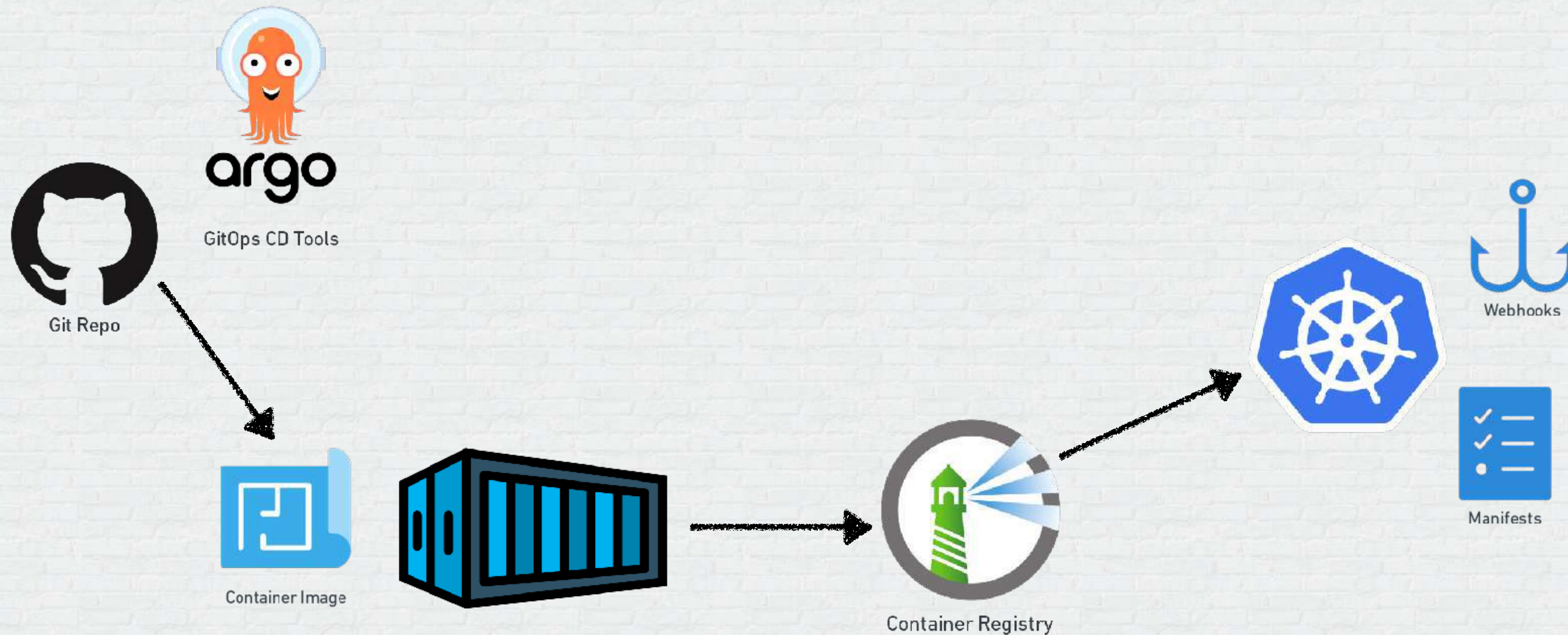


# Typical Players – Containers and K8s



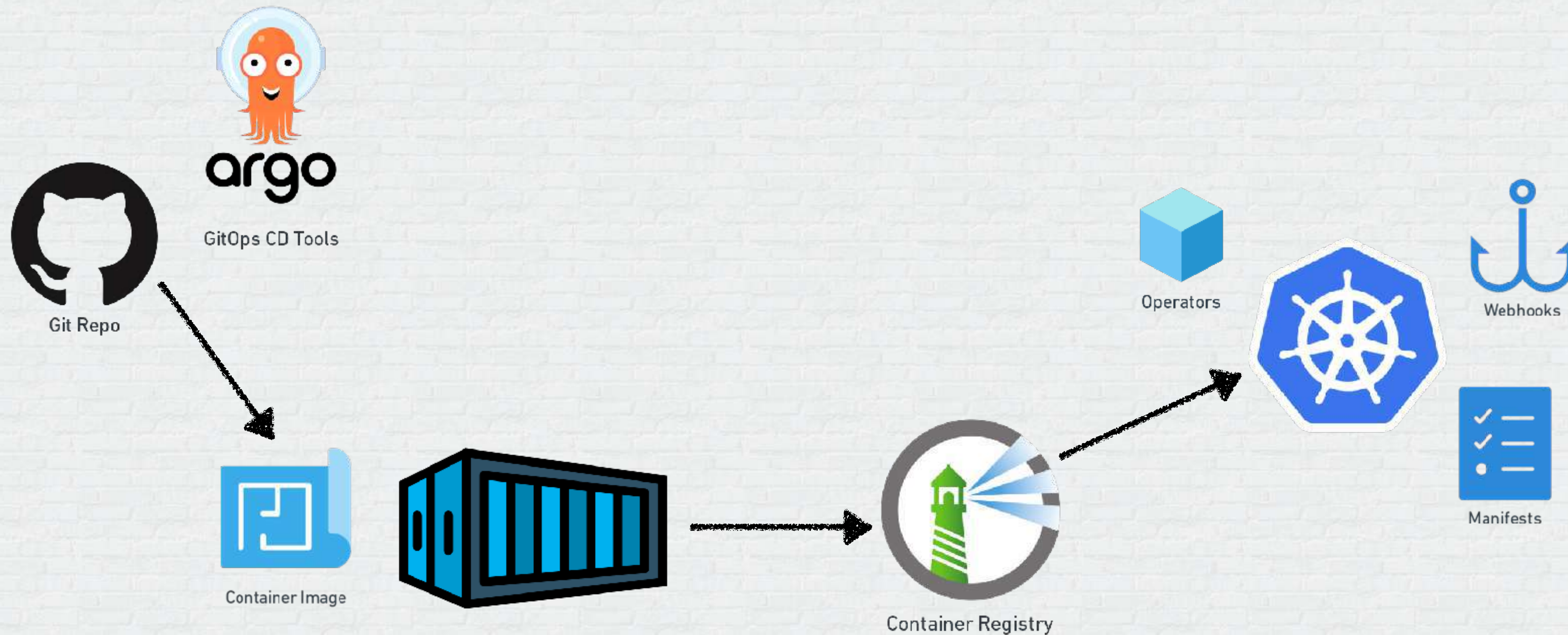


# Typical Players – Containers and K8s



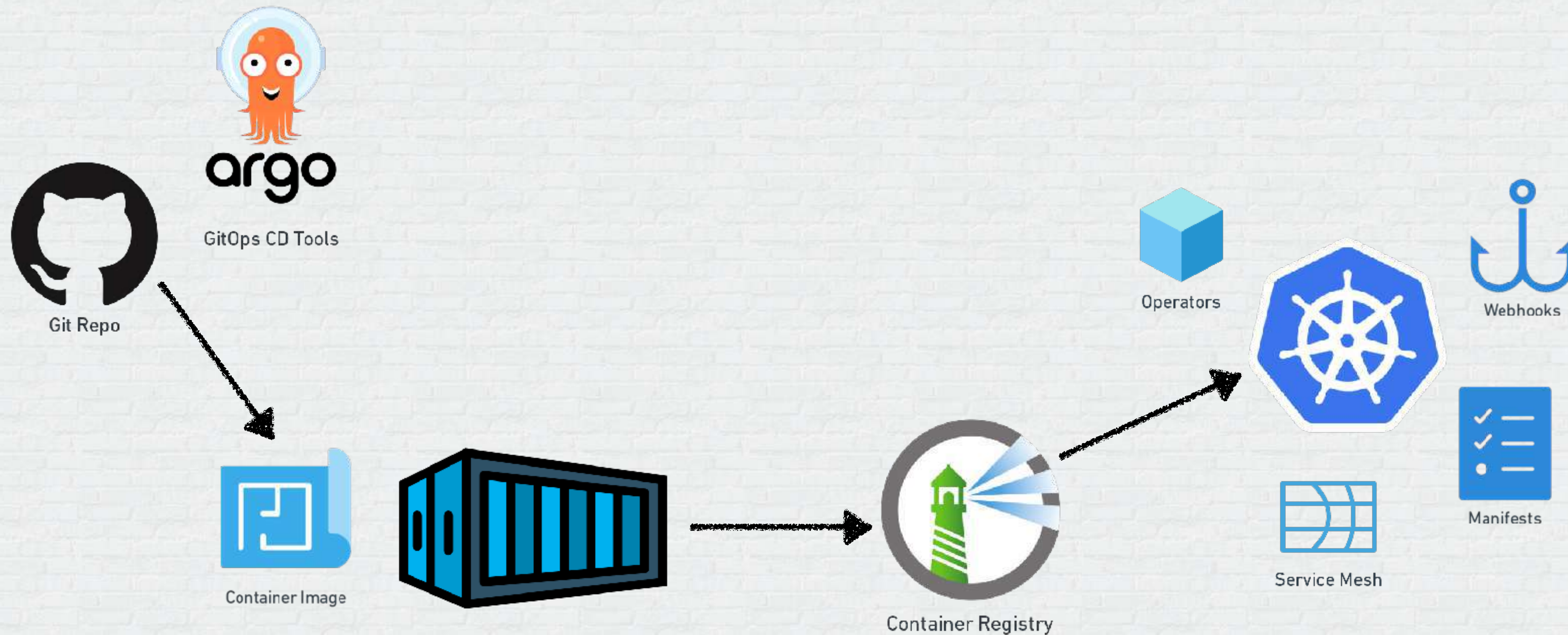


# Typical Players – Containers and K8s



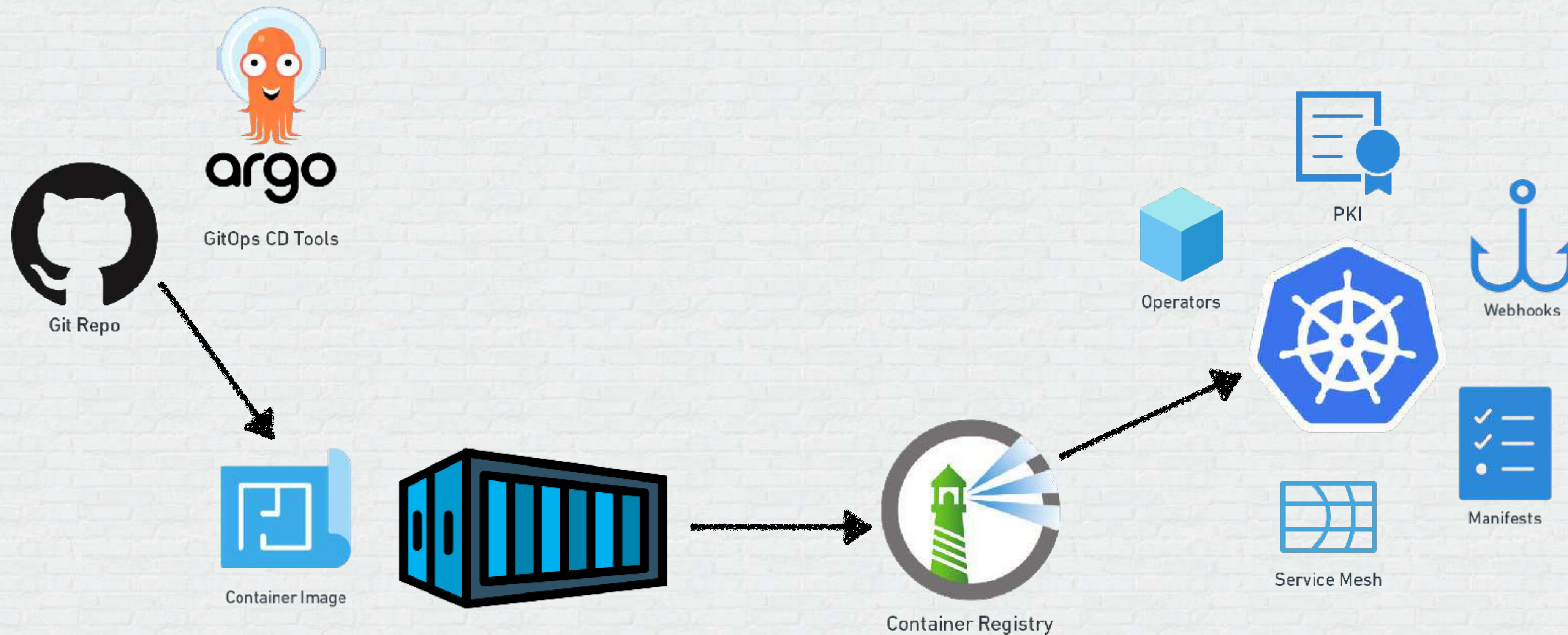


# Typical Players – Containers and K8s



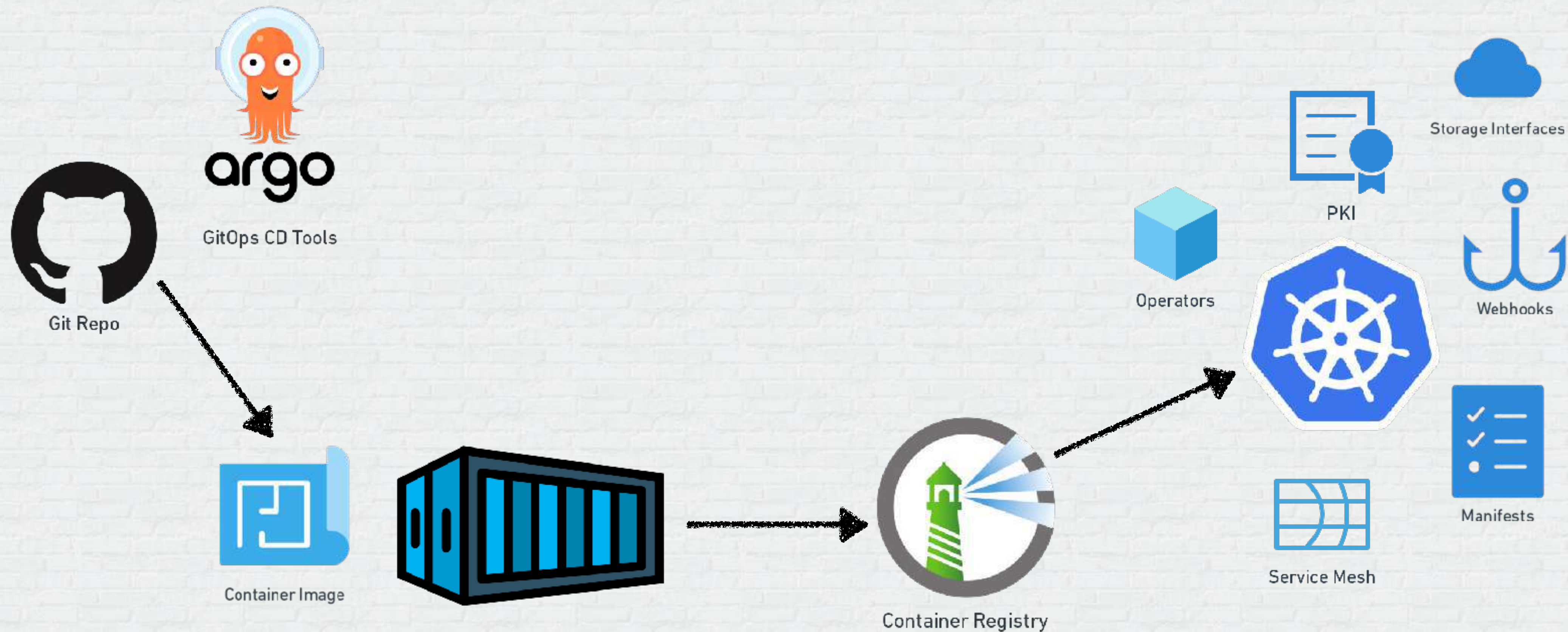


# Typical Players – Containers and K8s



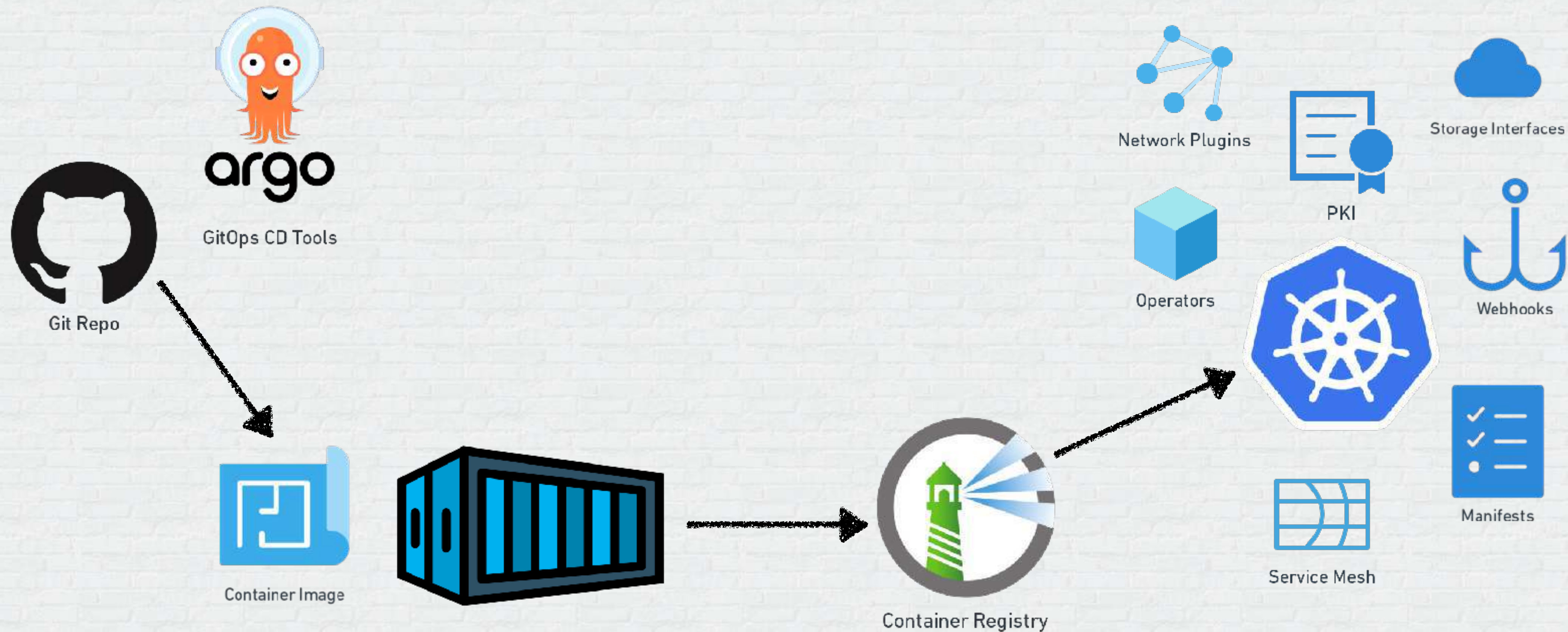


# Typical Players – Containers and K8s





# Typical Players – Containers and K8s





# Container Supply-Chain Security Considerations



GitOps CD Tools



# Container Supply-Chain Security Considerations





# Container Supply-Chain Security Considerations



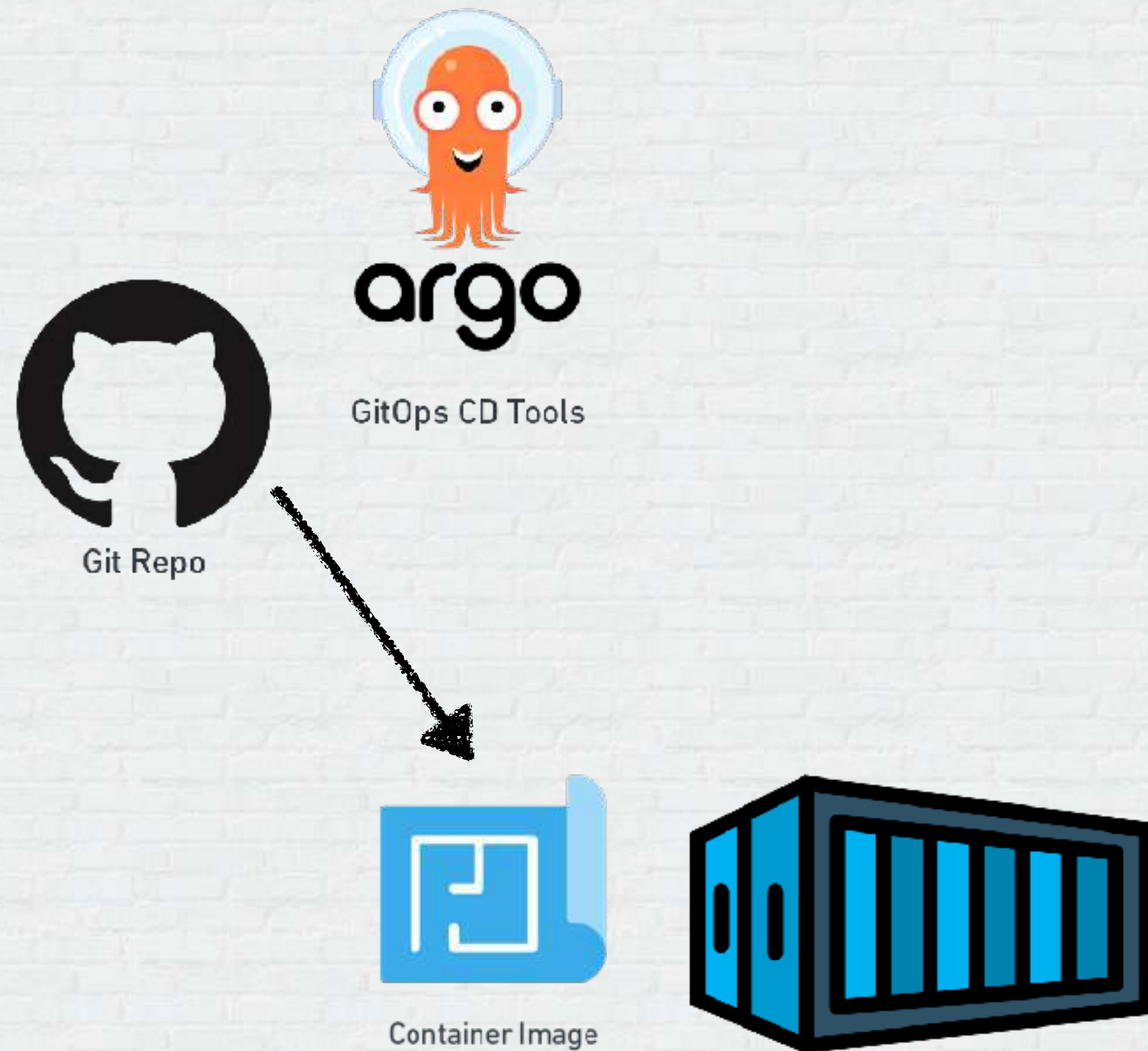


# Container Supply-Chain Security Considerations



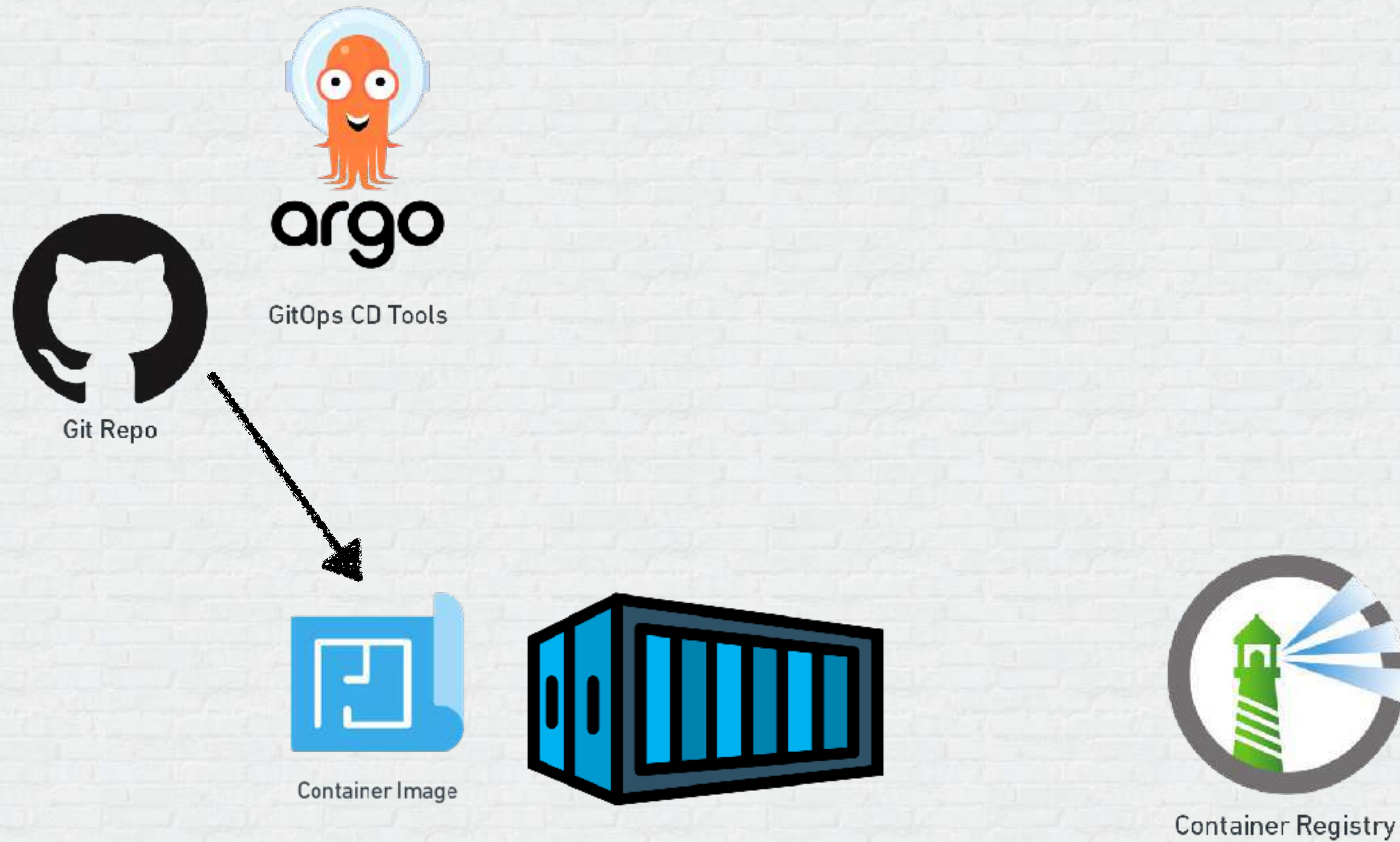


# Container Supply-Chain Security Considerations



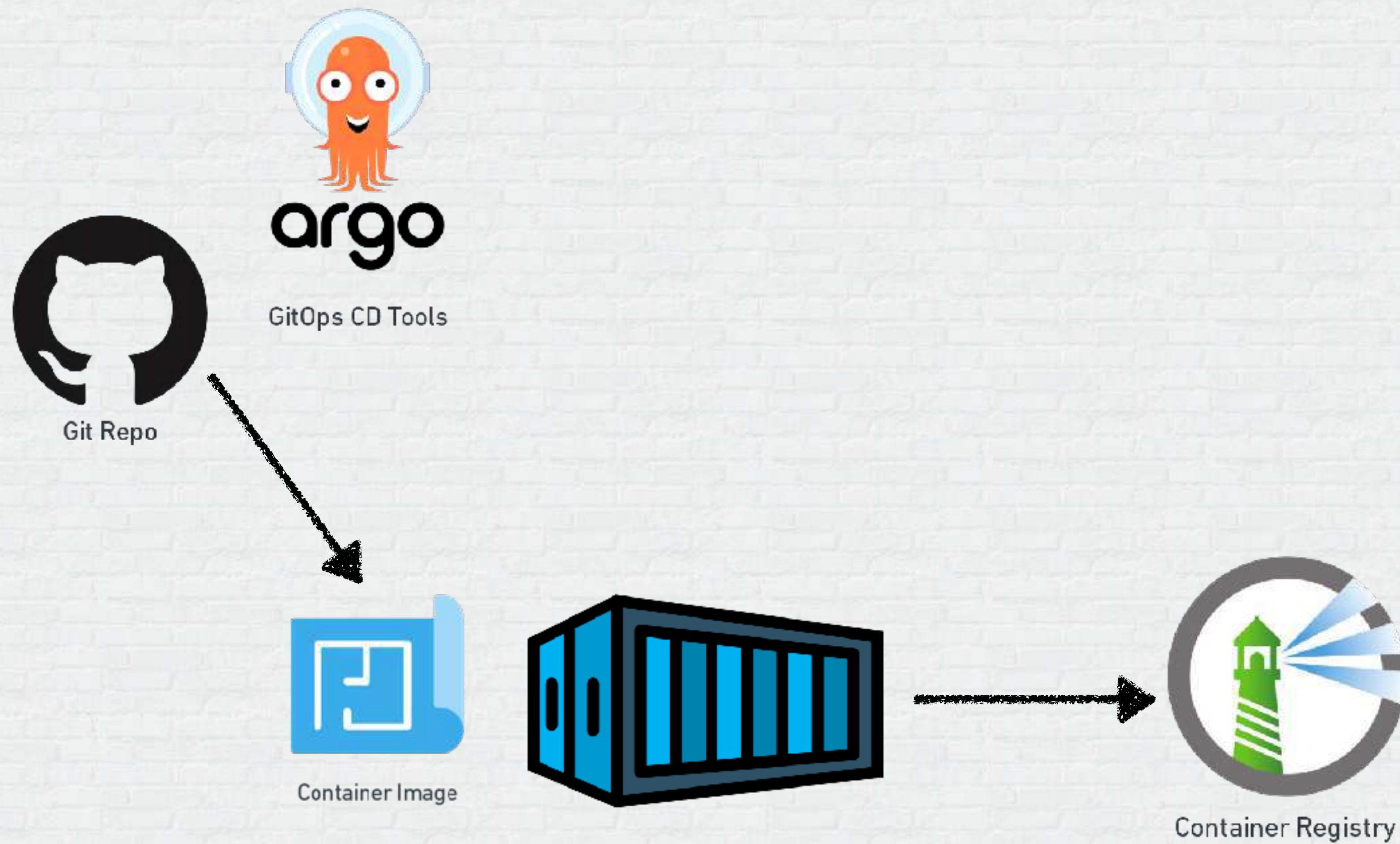


# Container Supply-Chain Security Considerations



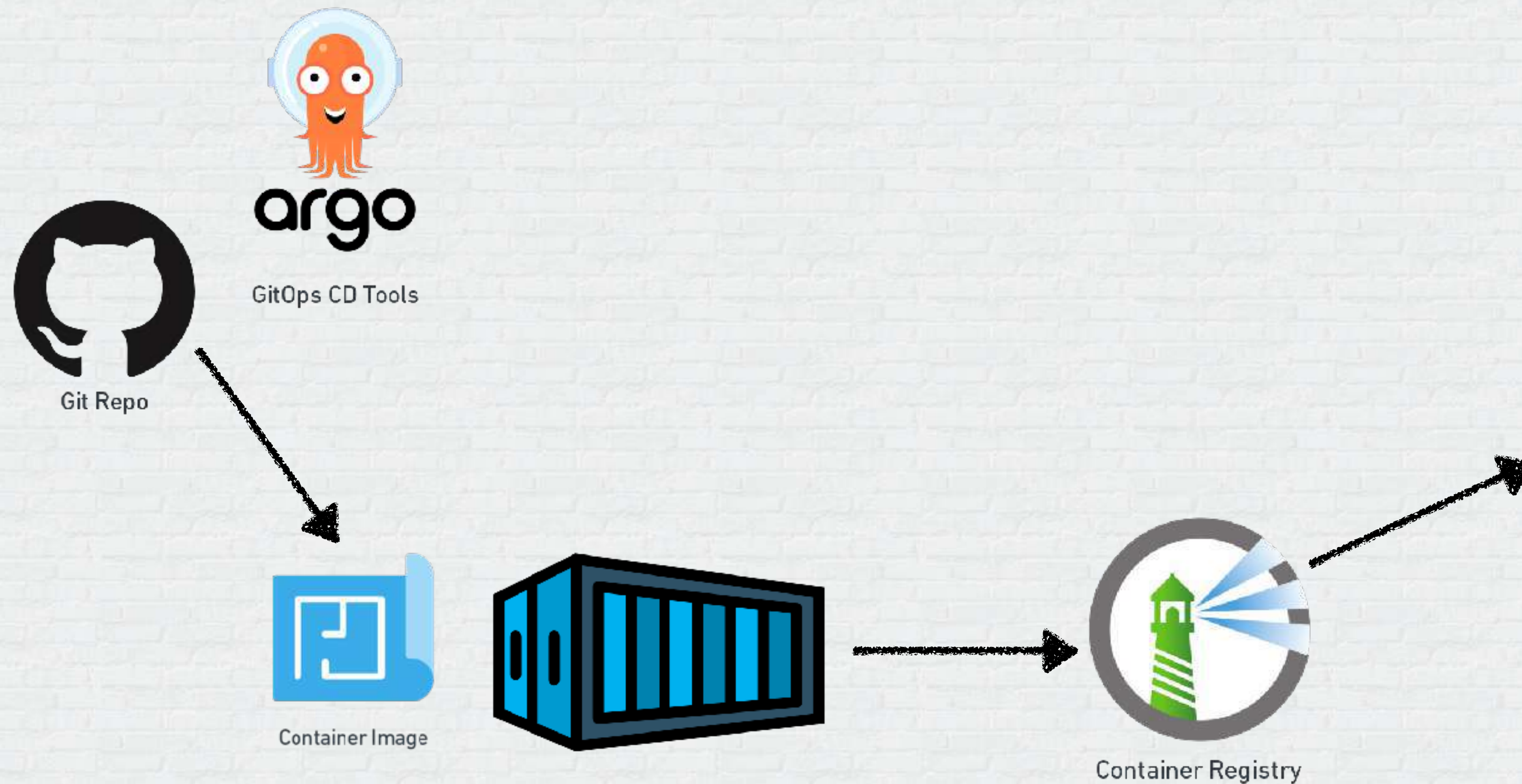


# Container Supply-Chain Security Considerations



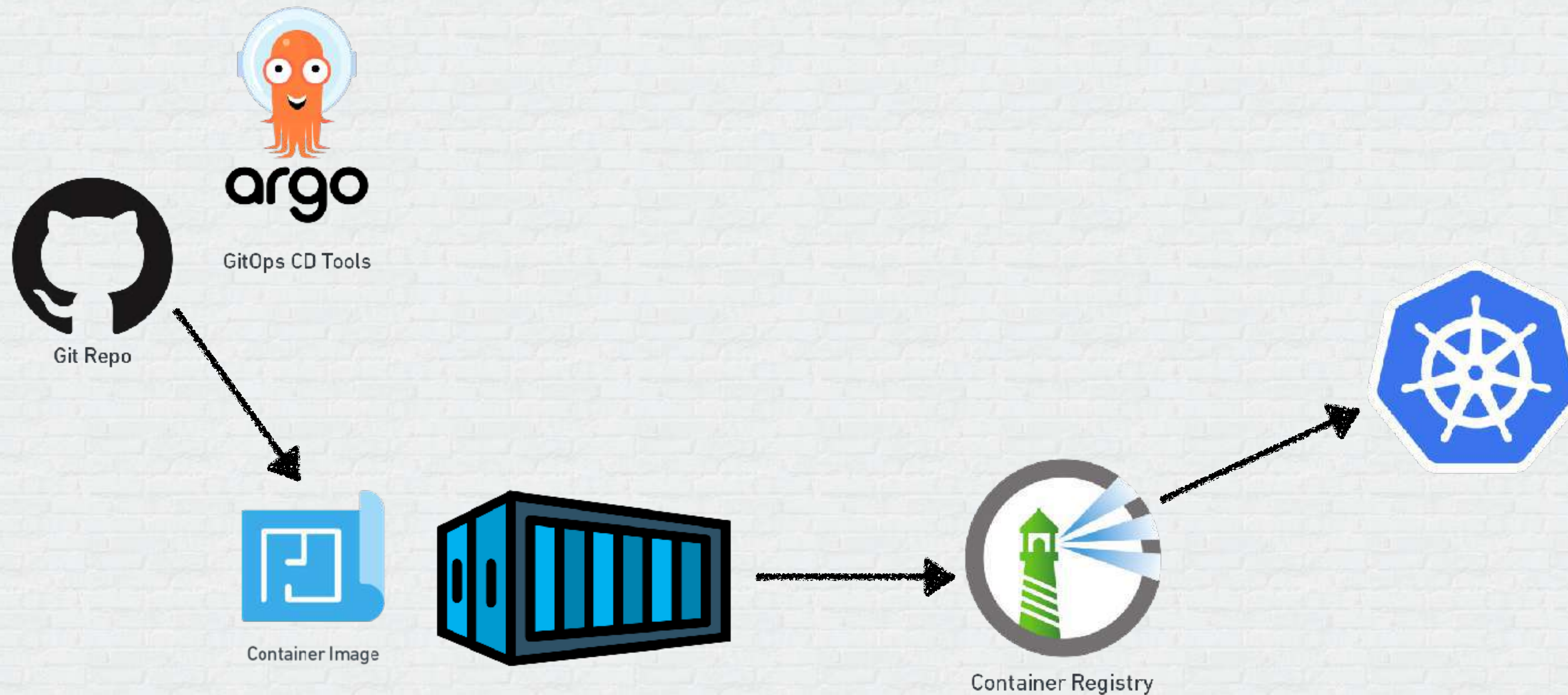


# Container Supply-Chain Security Considerations



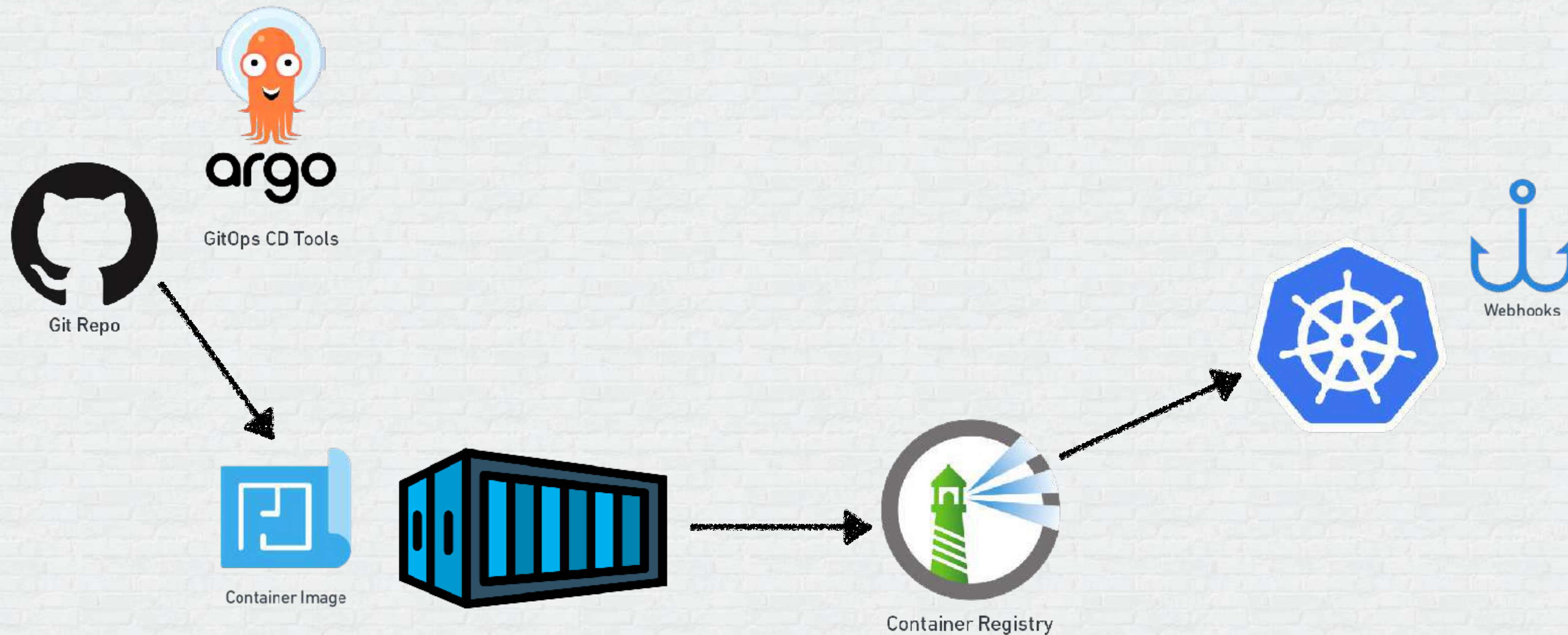


# Container Supply-Chain Security Considerations



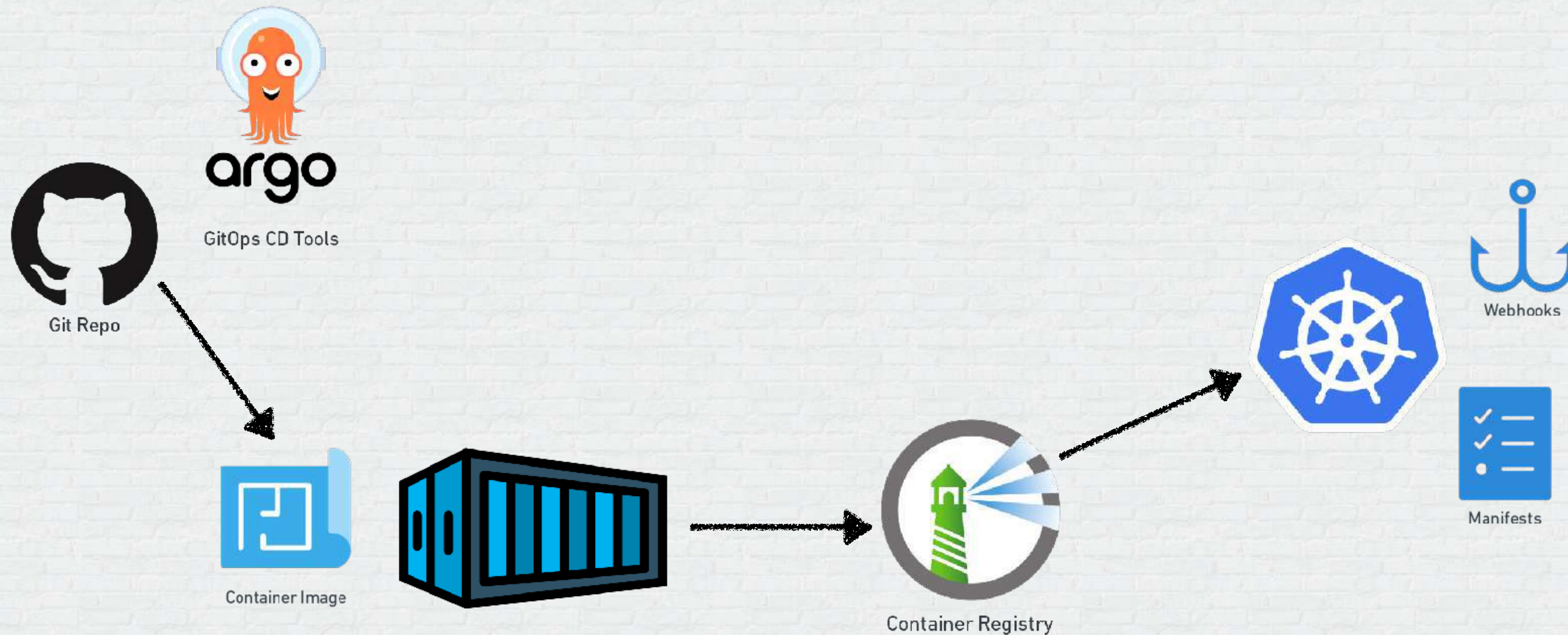


# Container Supply-Chain Security Considerations



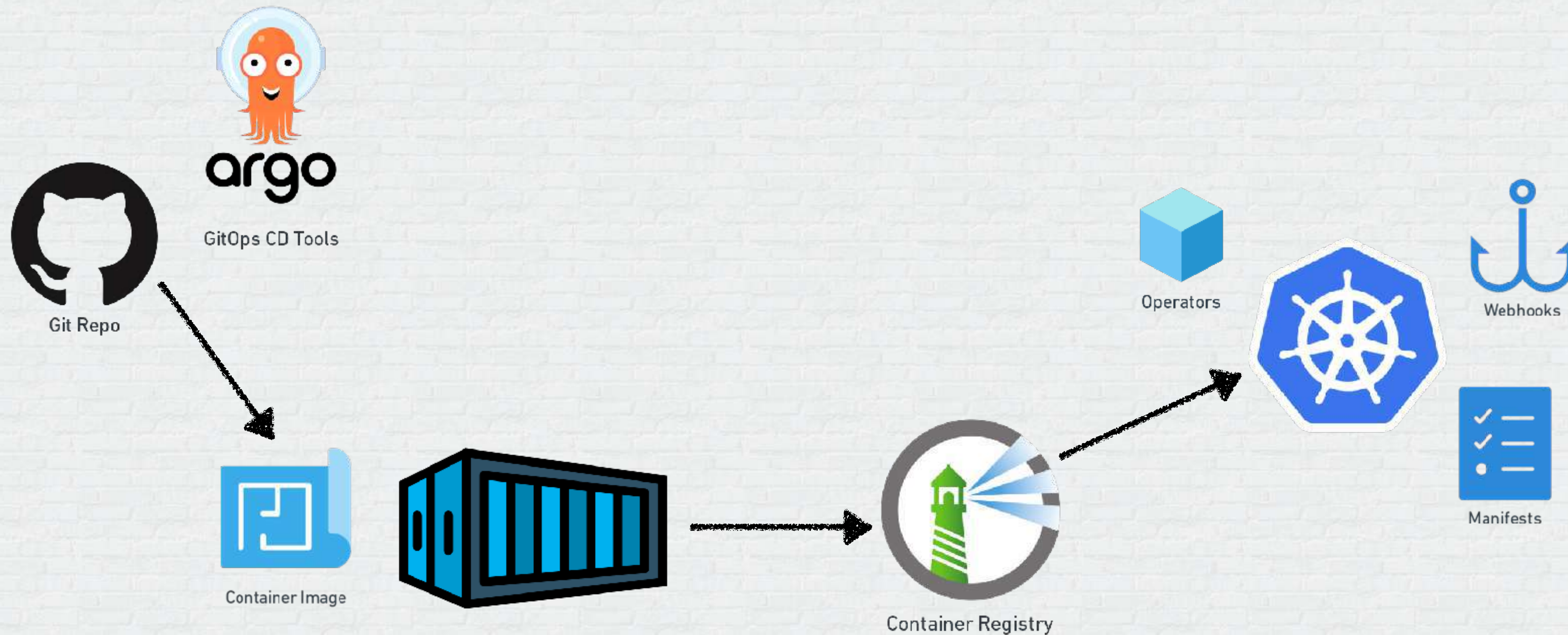


# Container Supply-Chain Security Considerations



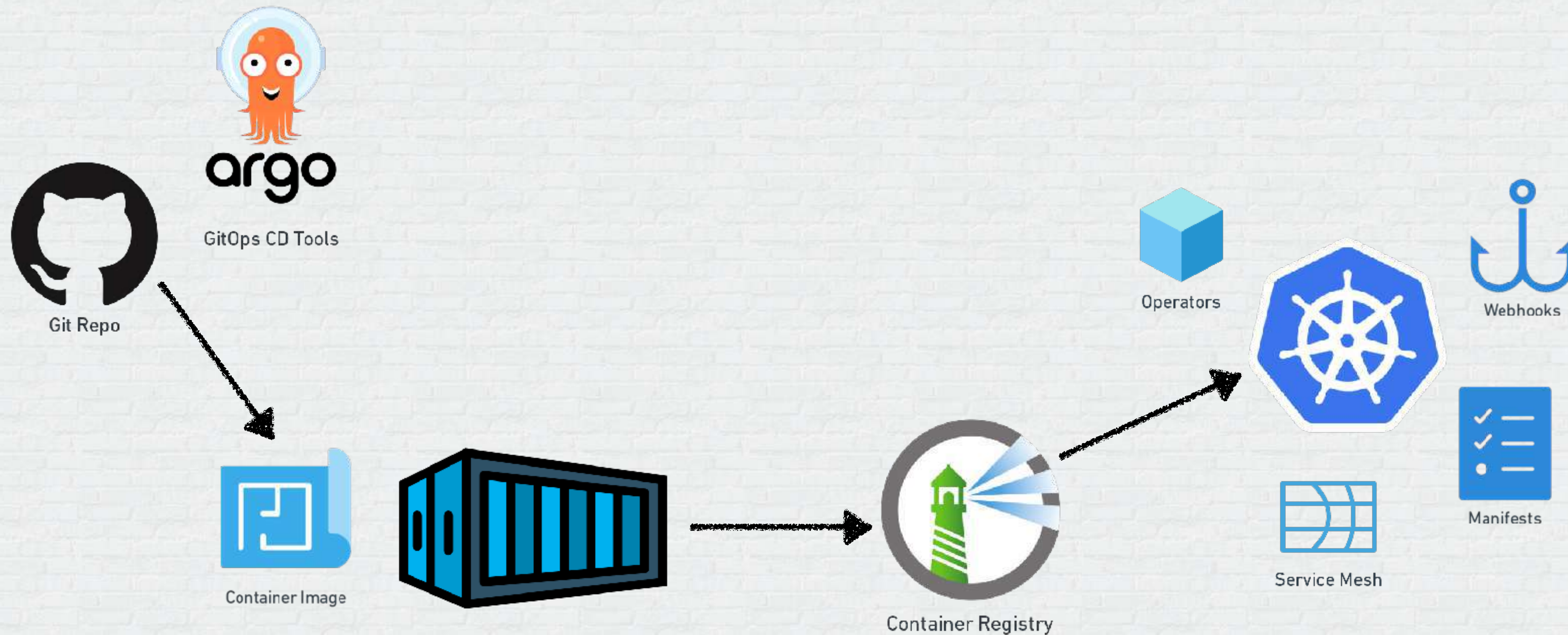


# Container Supply-Chain Security Considerations



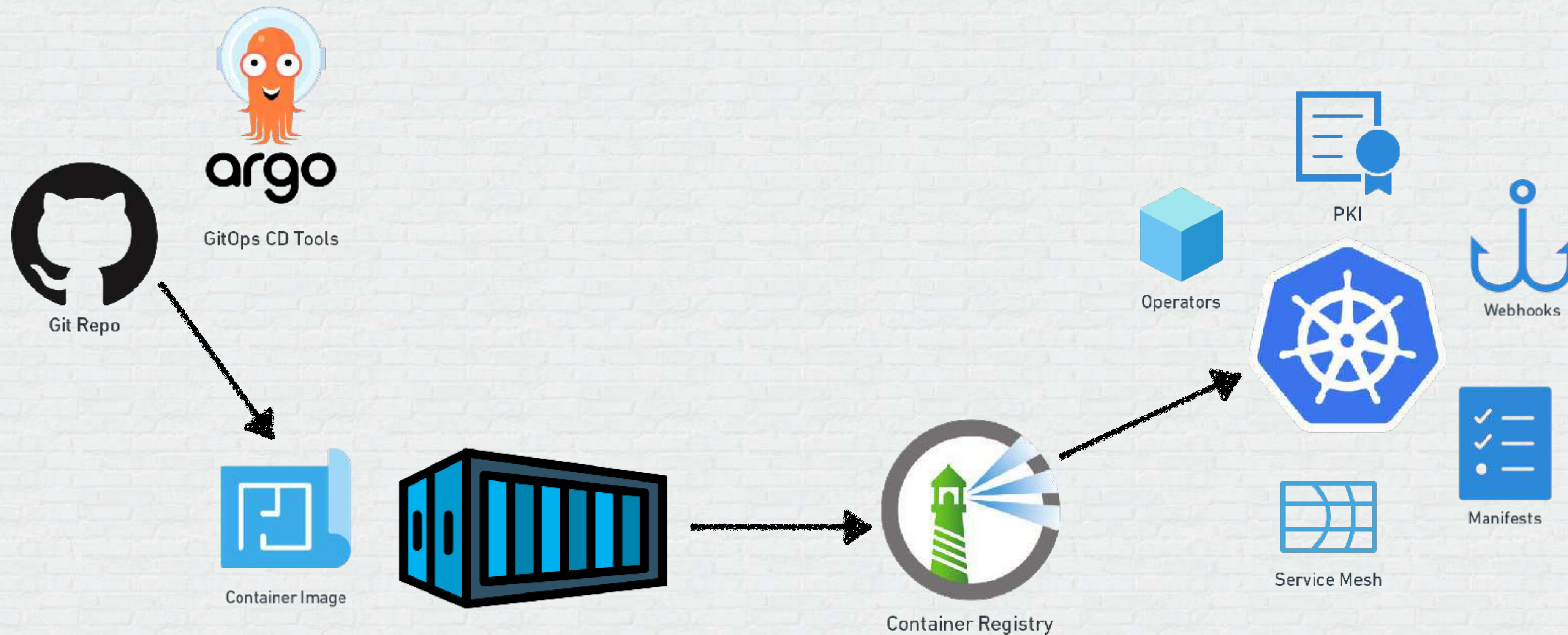


# Container Supply-Chain Security Considerations



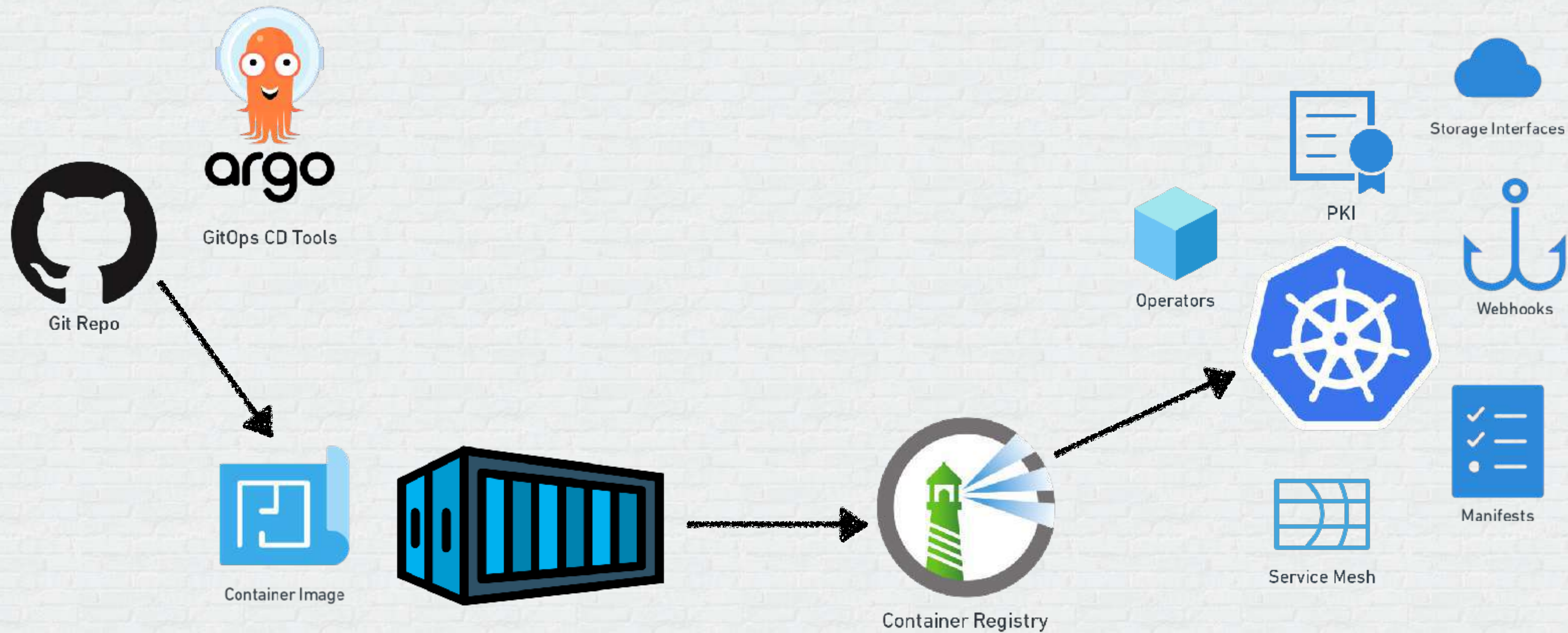


# Container Supply-Chain Security Considerations



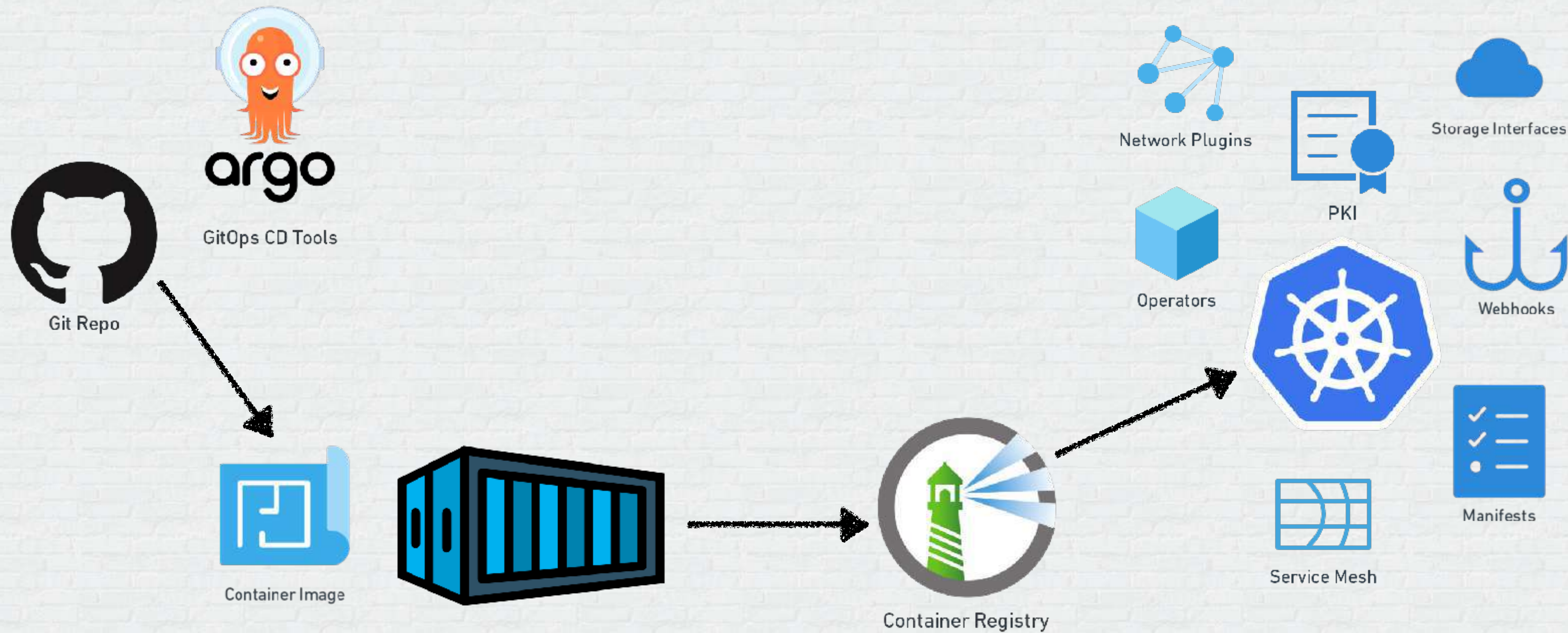


# Container Supply-Chain Security Considerations



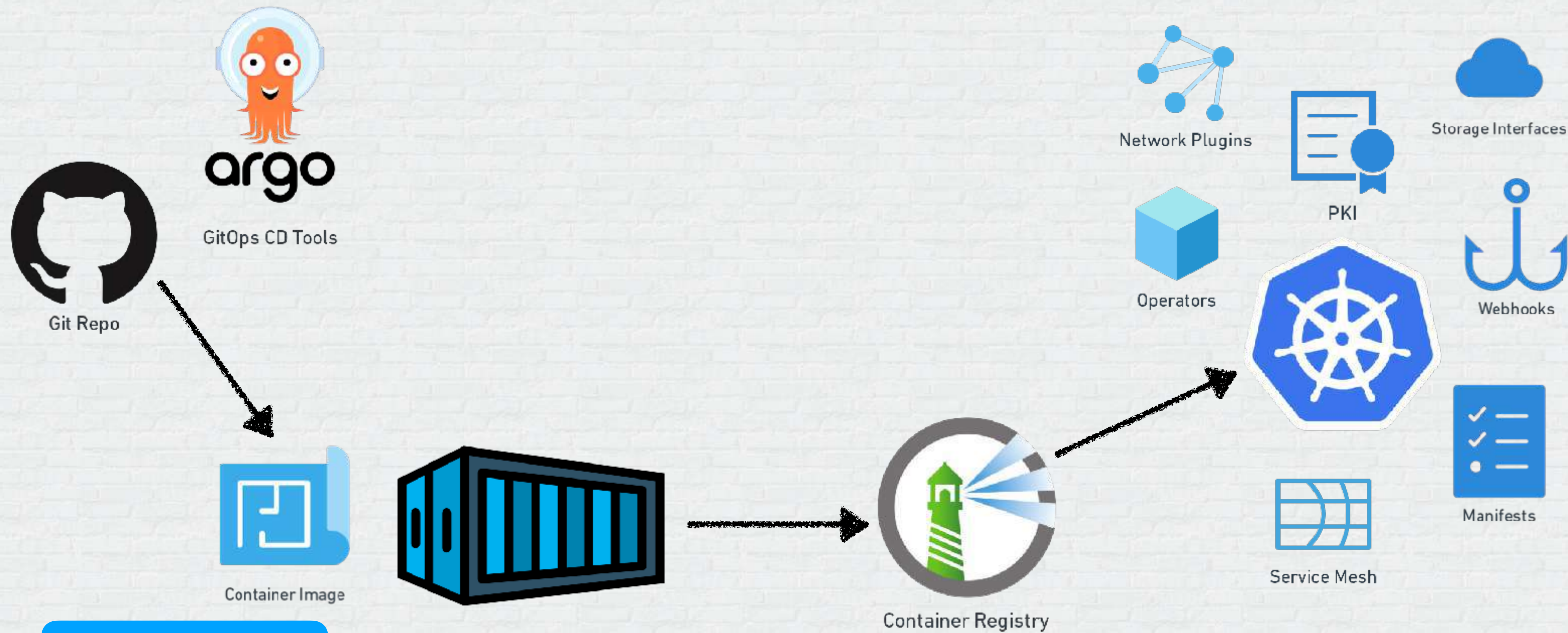


# Container Supply-Chain Security Considerations





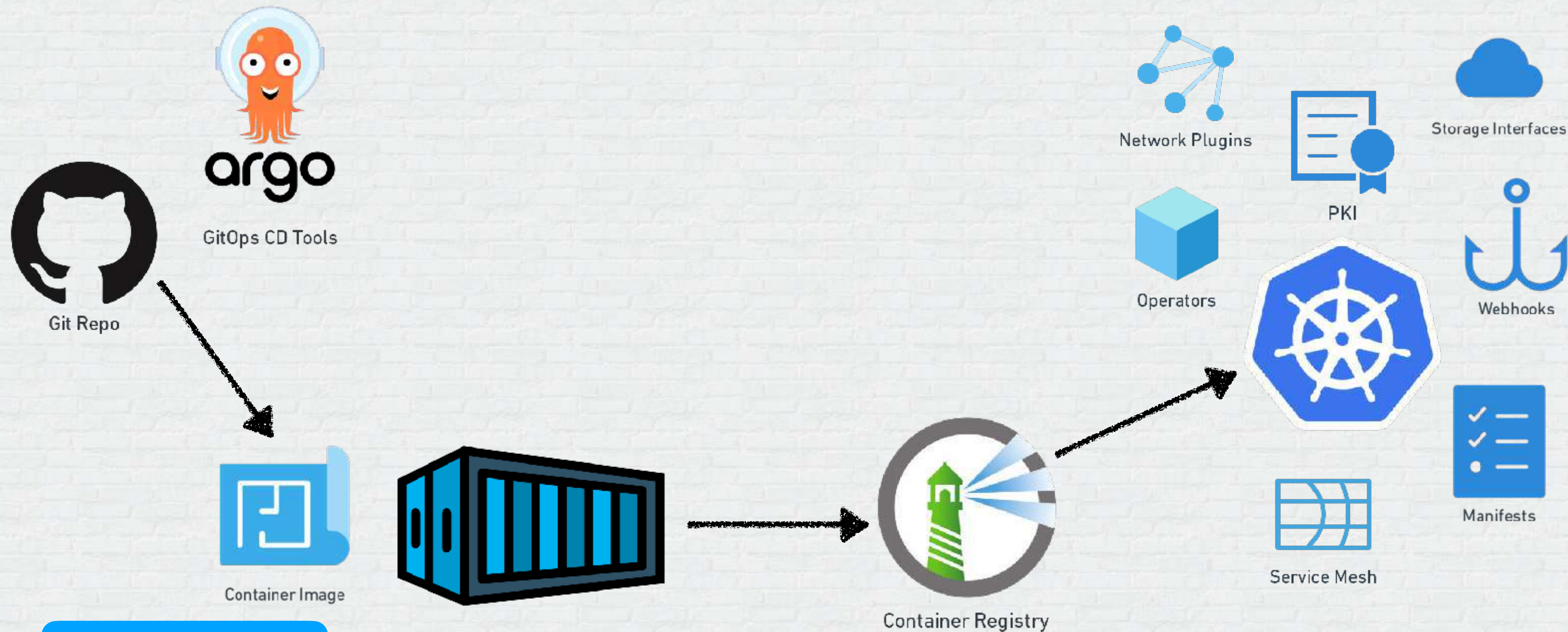
# Container Supply-Chain Security Considerations



- Code in layers



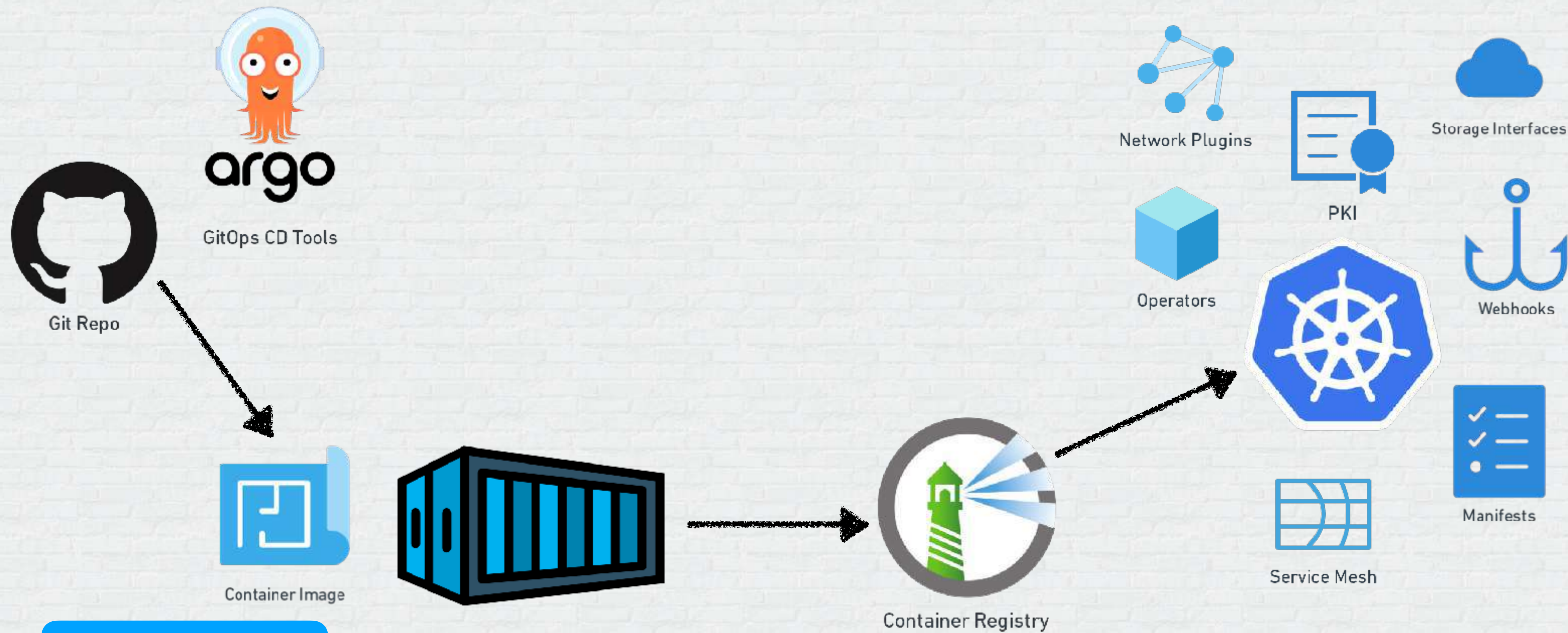
# Container Supply-Chain Security Considerations



- Code in layers
- Base Image



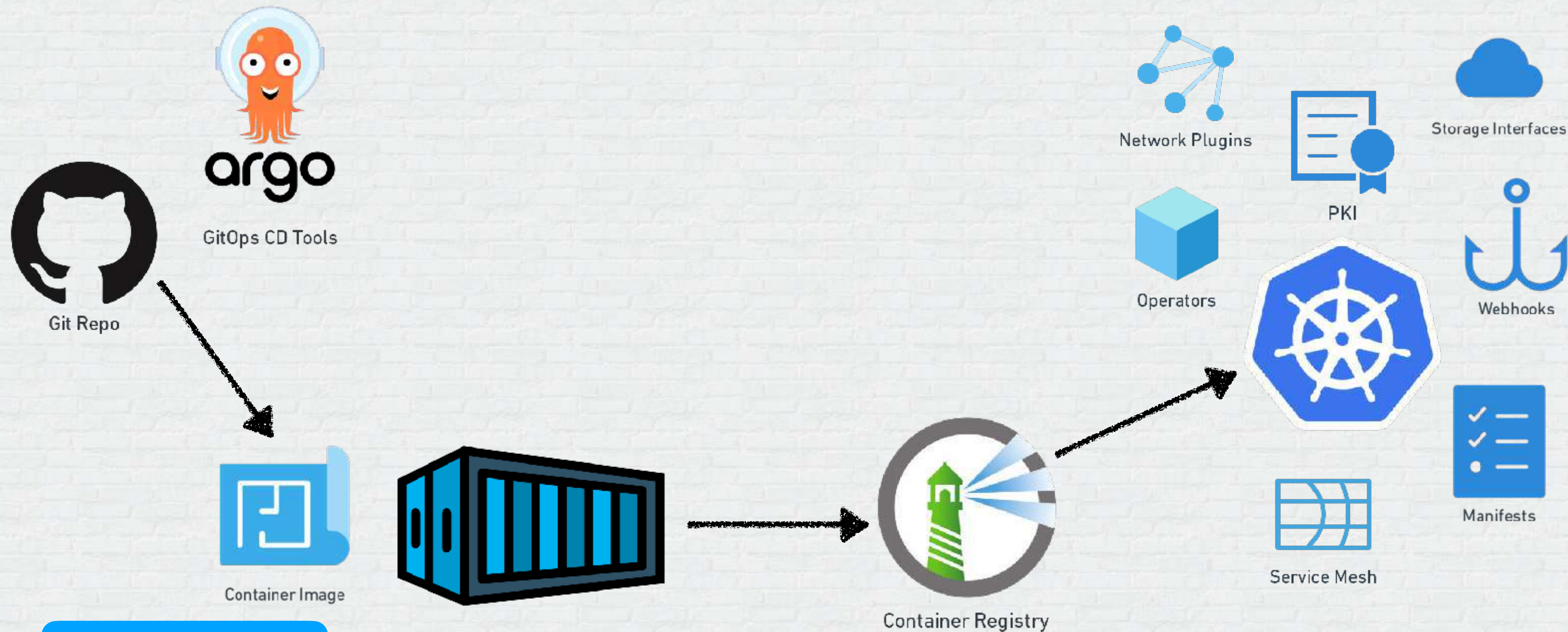
# Container Supply-Chain Security Considerations



- Code in layers
- Base Image
- Secrets



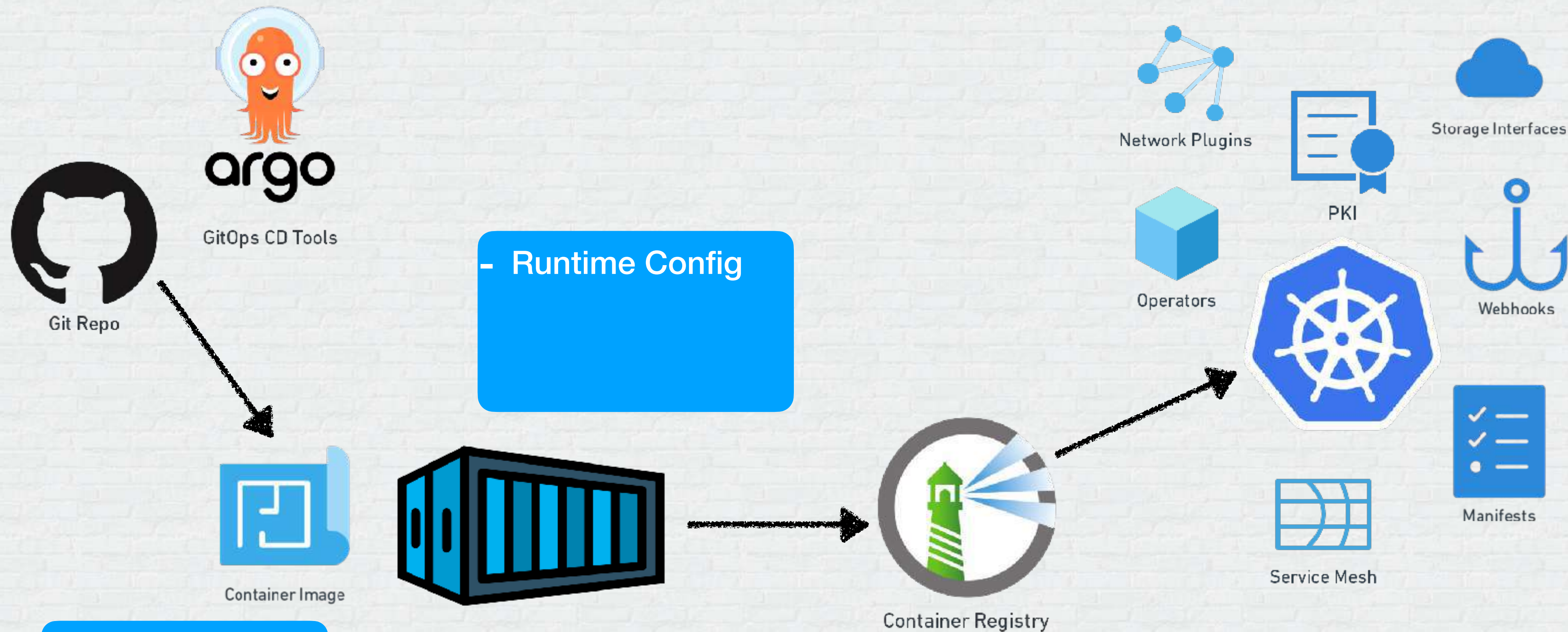
# Container Supply-Chain Security Considerations



- Code in layers
- Base Image
- Secrets
- AuthN



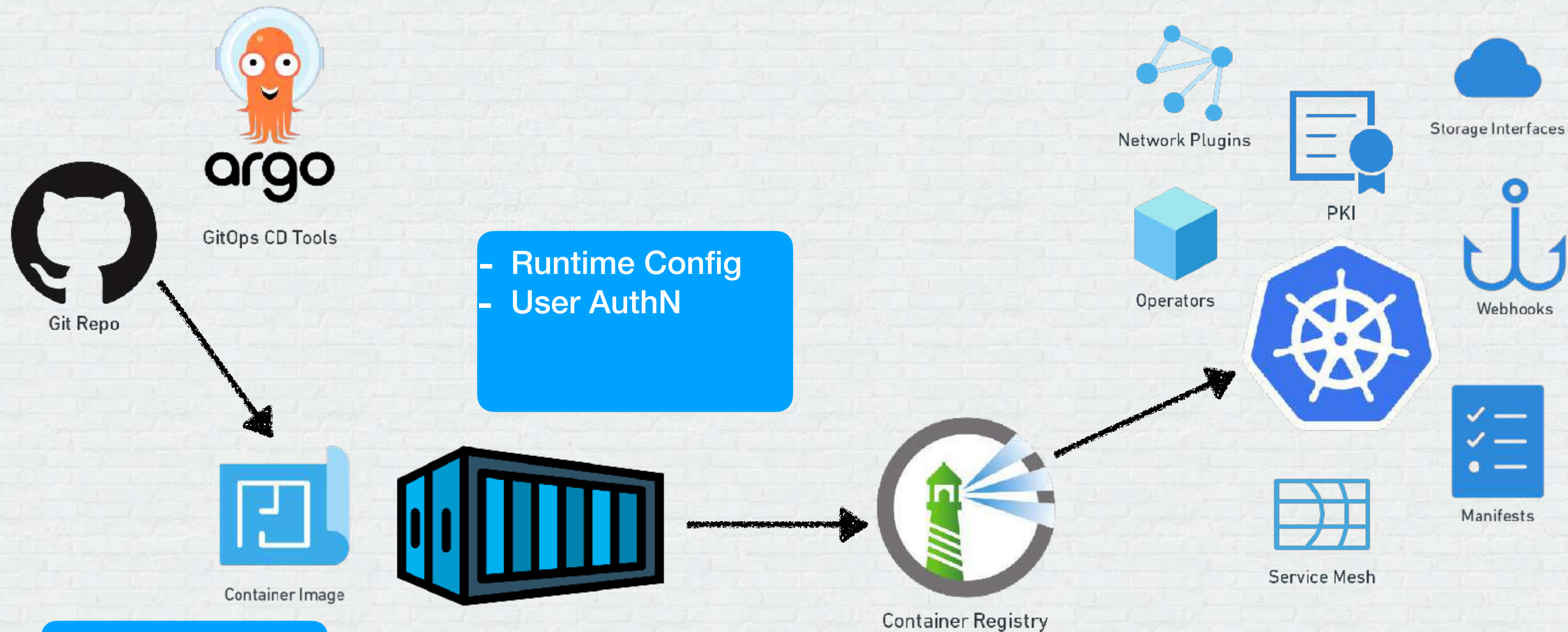
# Container Supply-Chain Security Considerations



- Code in layers
- Base Image
- Secrets
- AuthN



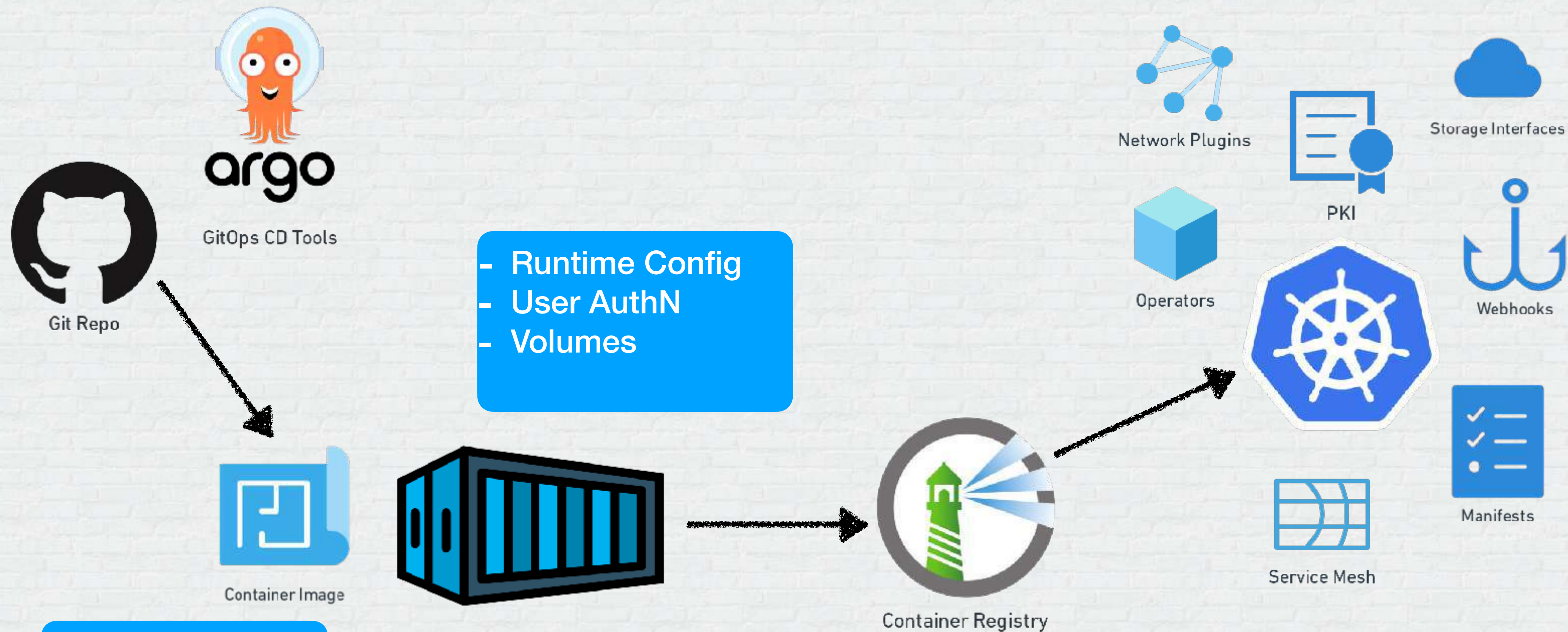
# Container Supply-Chain Security Considerations



- Code in layers
- Base Image
- Secrets
- AuthN

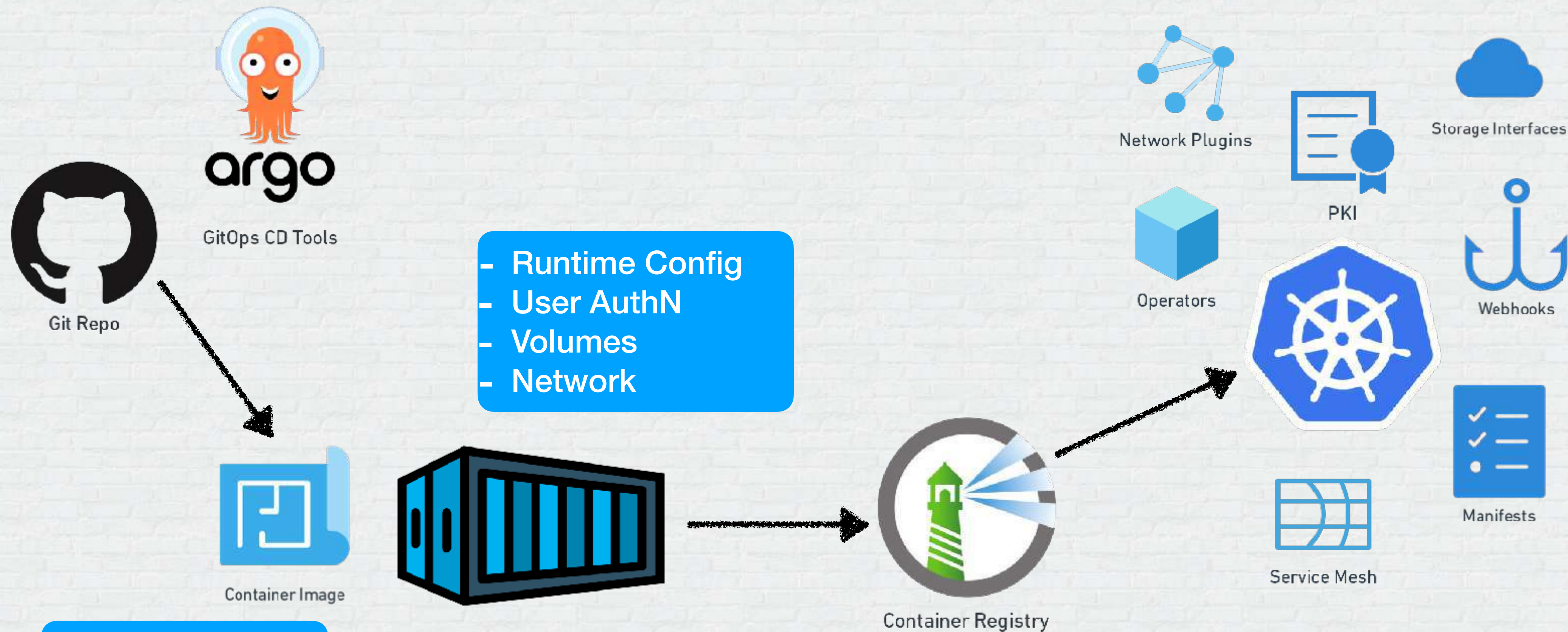


# Container Supply-Chain Security Considerations





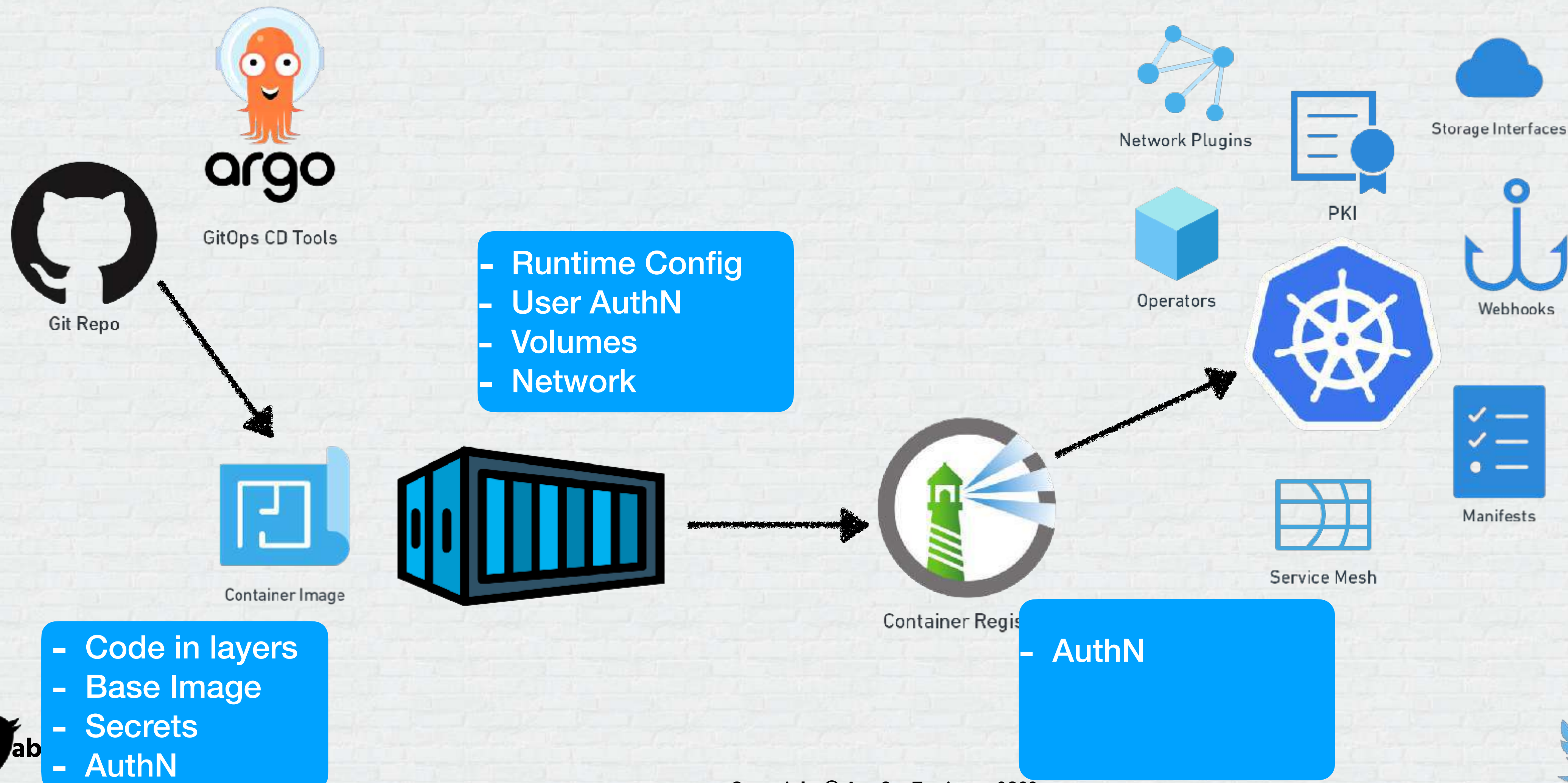
# Container Supply-Chain Security Considerations



- Code in layers
- Base Image
- Secrets
- AuthN

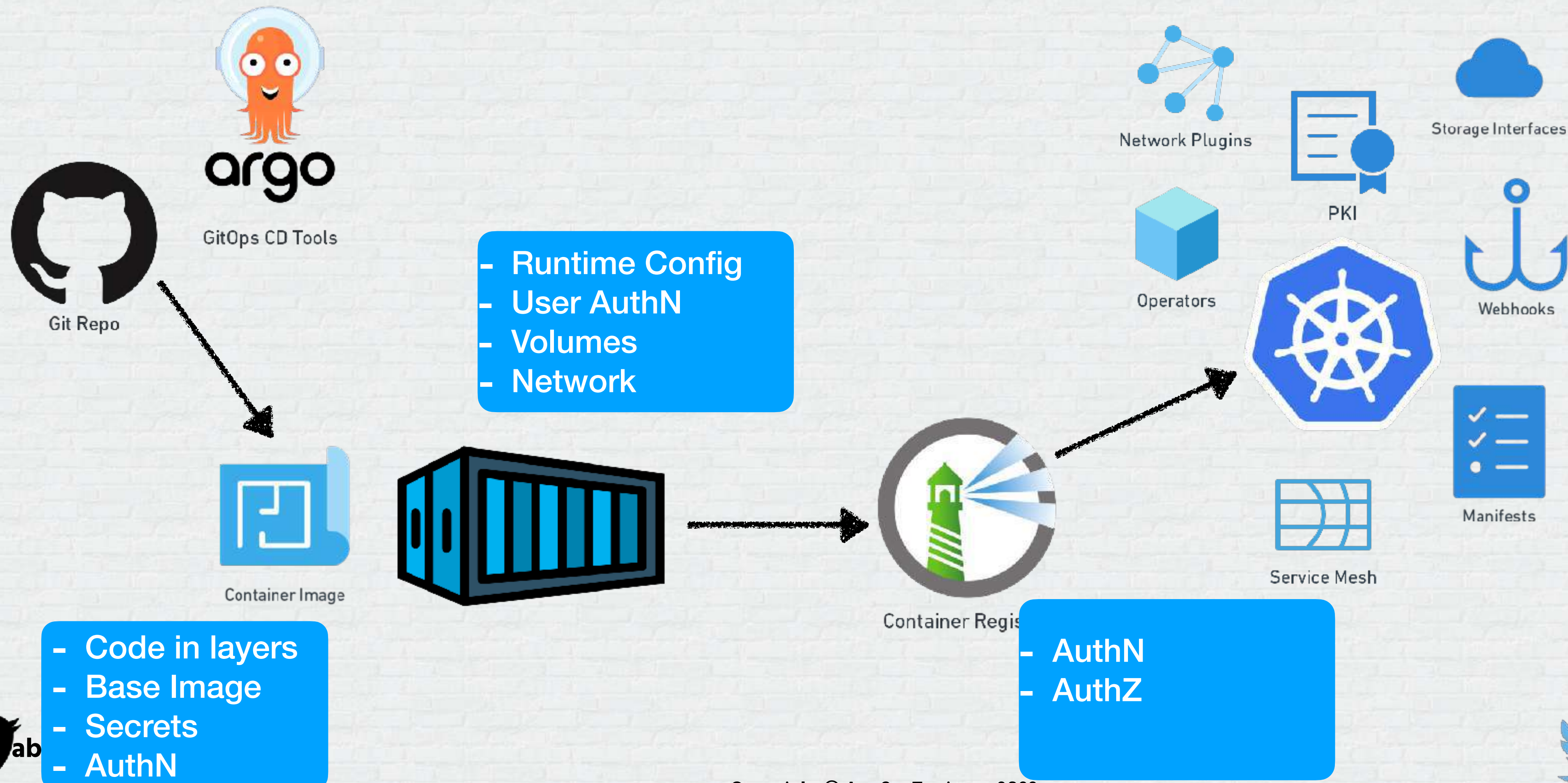


# Container Supply-Chain Security Considerations



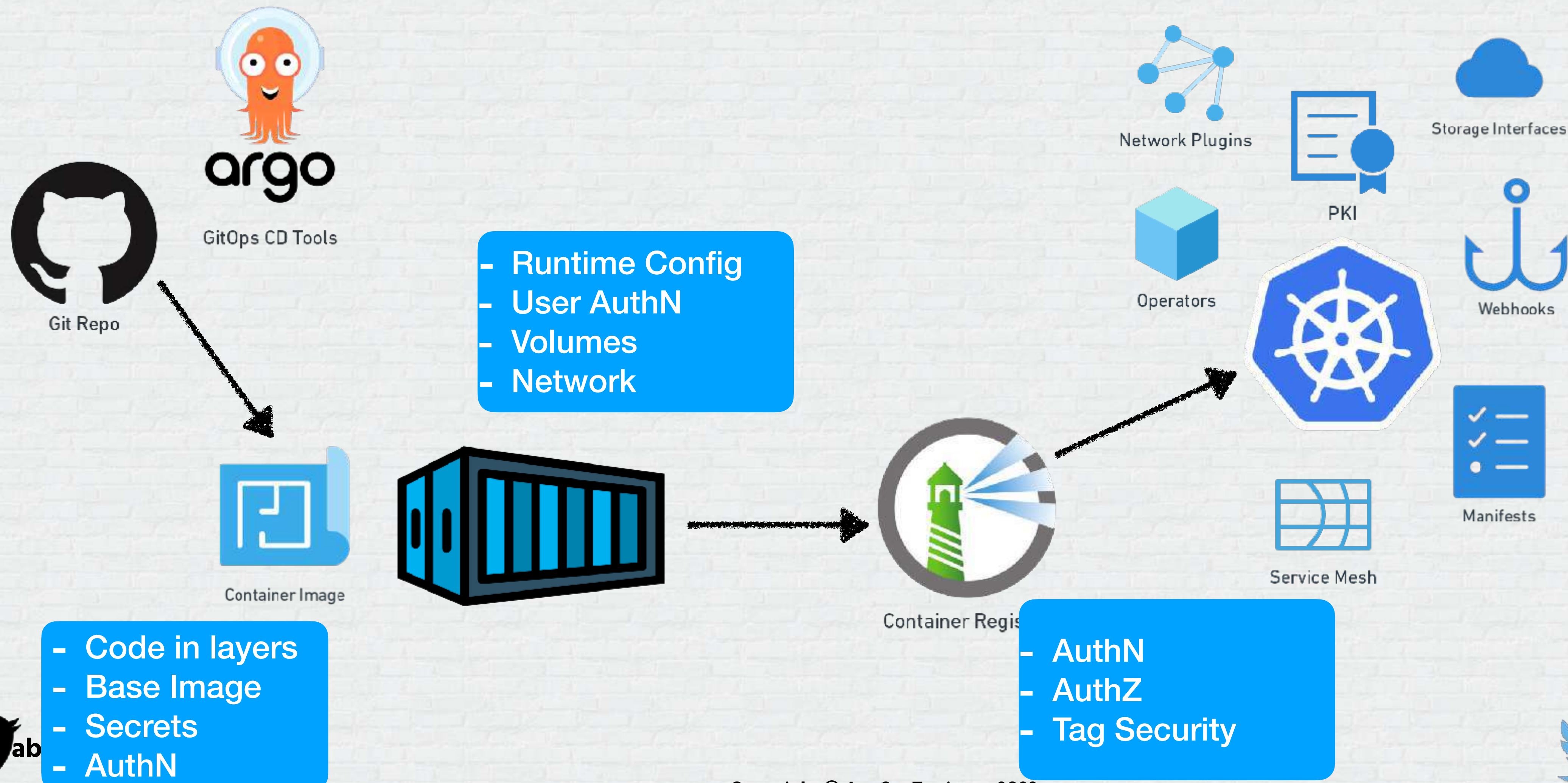


# Container Supply-Chain Security Considerations





# Container Supply-Chain Security Considerations





# **\$1 Tour of Kubernetes Admission Controllers**





# Admission Control – K8s



# Validating Web Hook



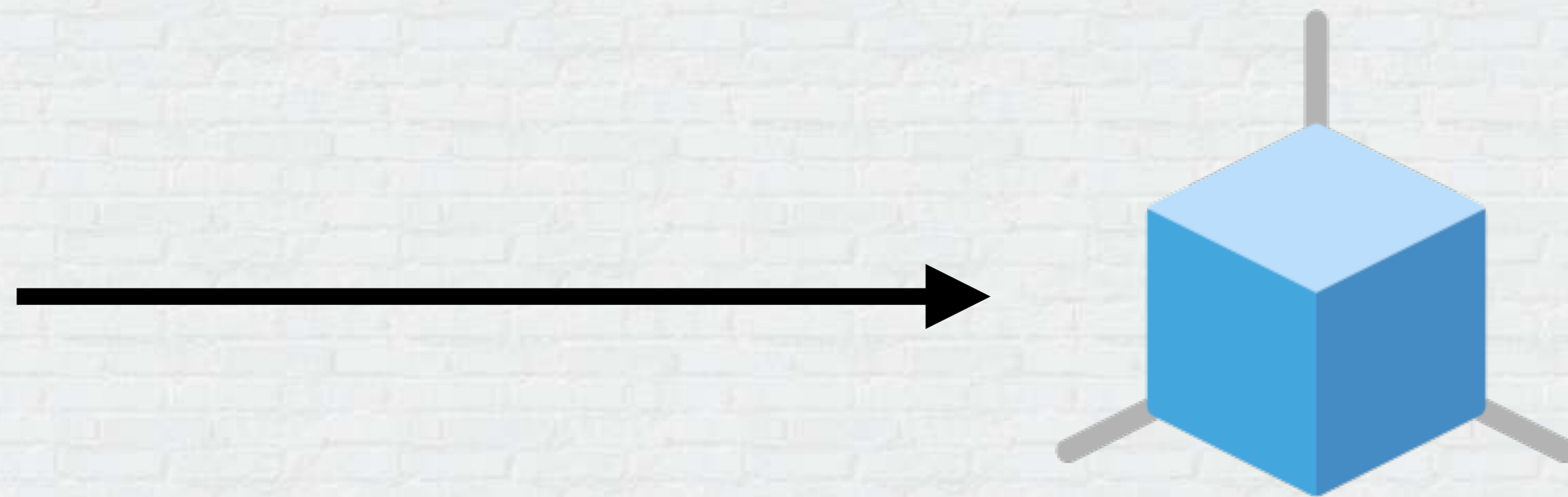
# Validating Web Hook





# Validating Web Hook

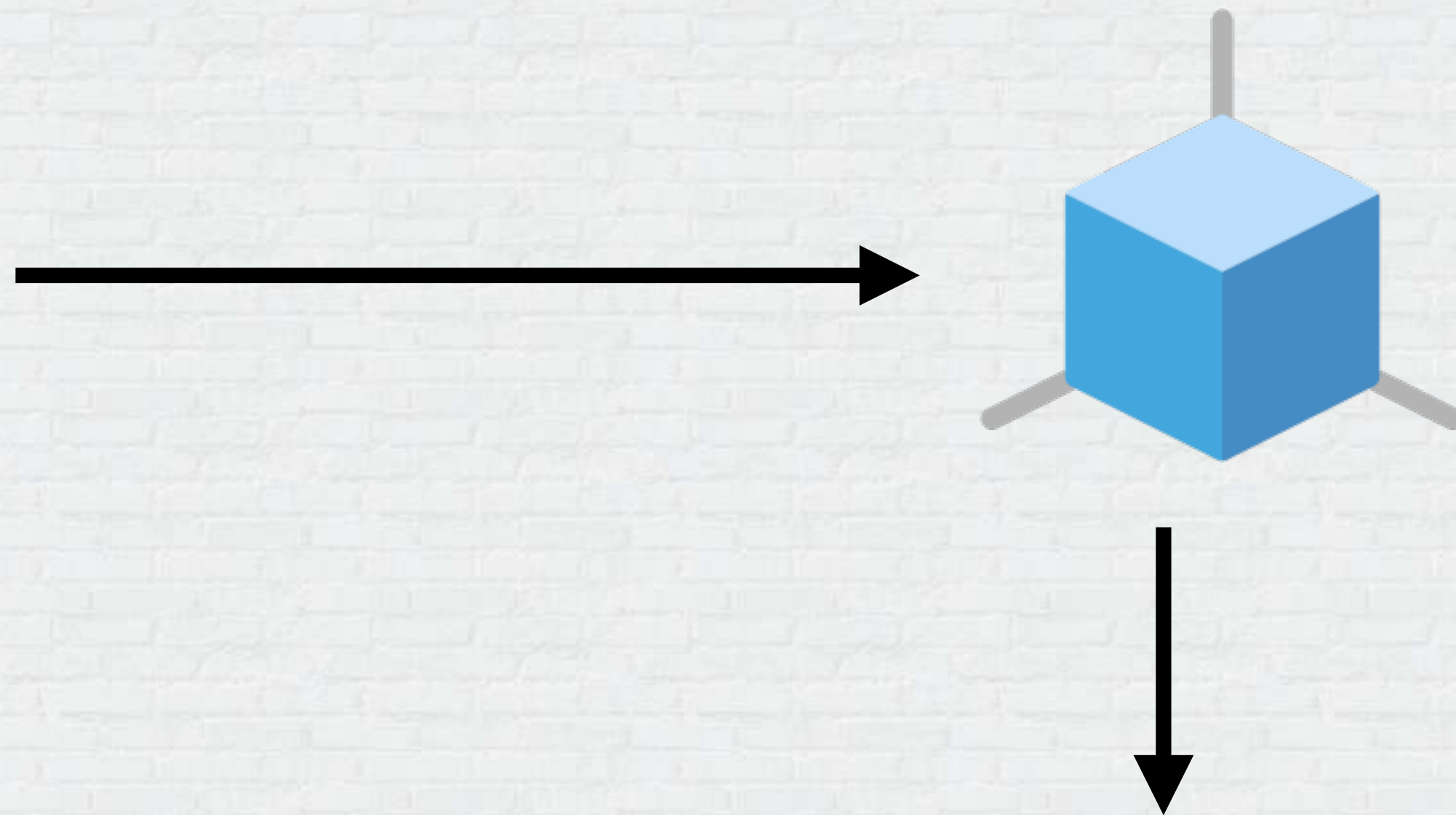
Validation Admission Controller





# Validating Web Hook

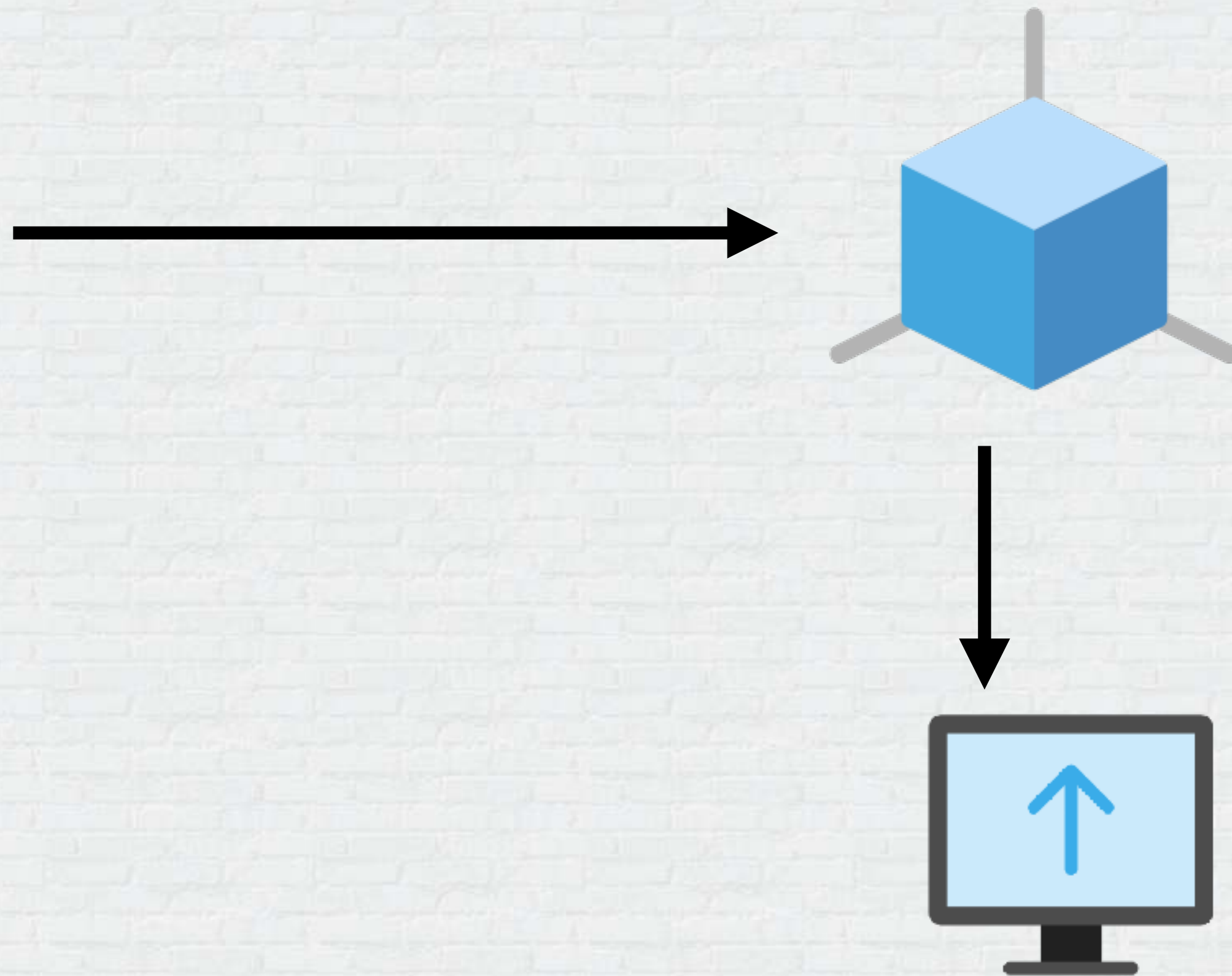
Validation Admission Controller





# Validating Web Hook

Validation Admission Controller

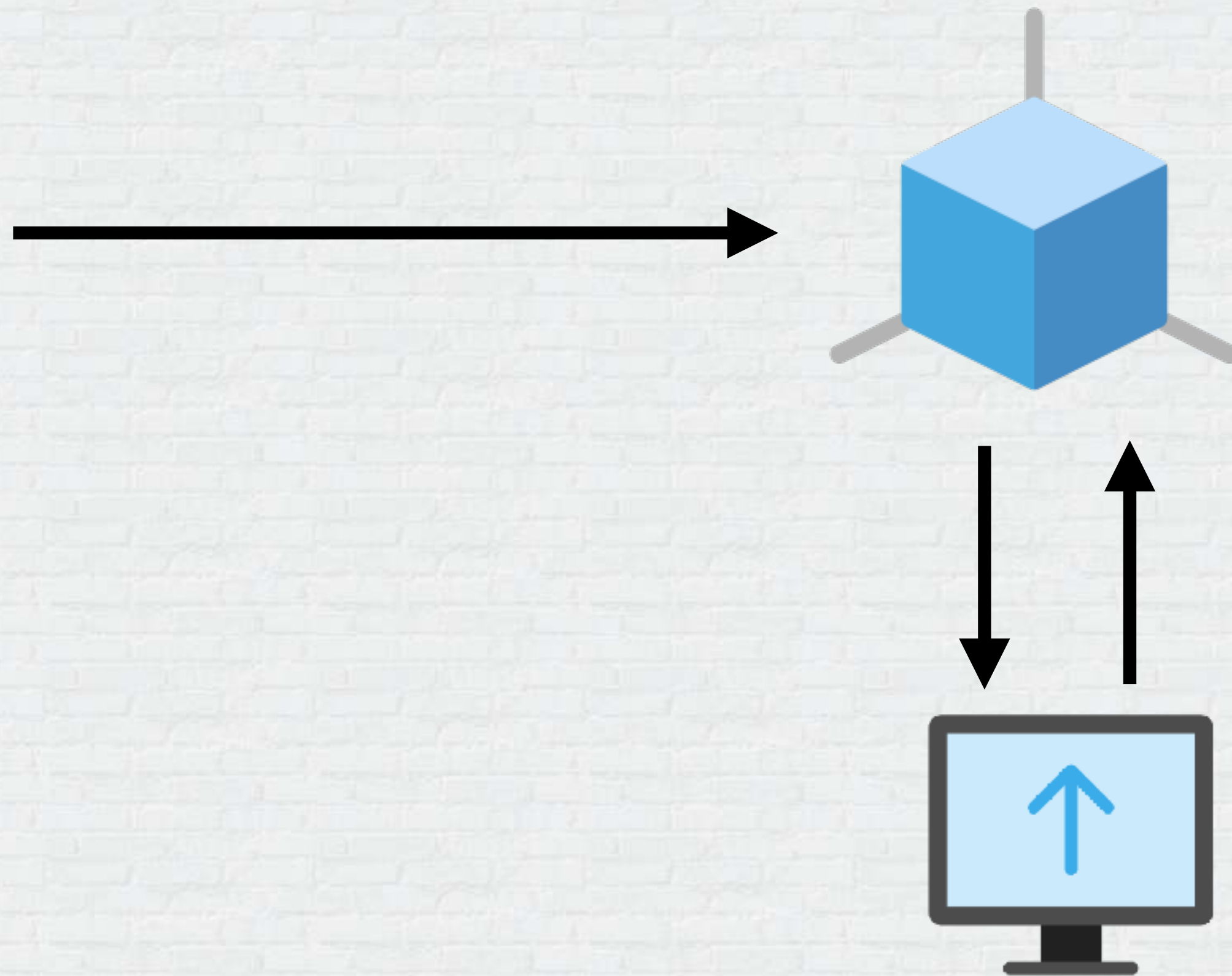


Validating Webhook



# Validating Web Hook

Validation Admission Controller

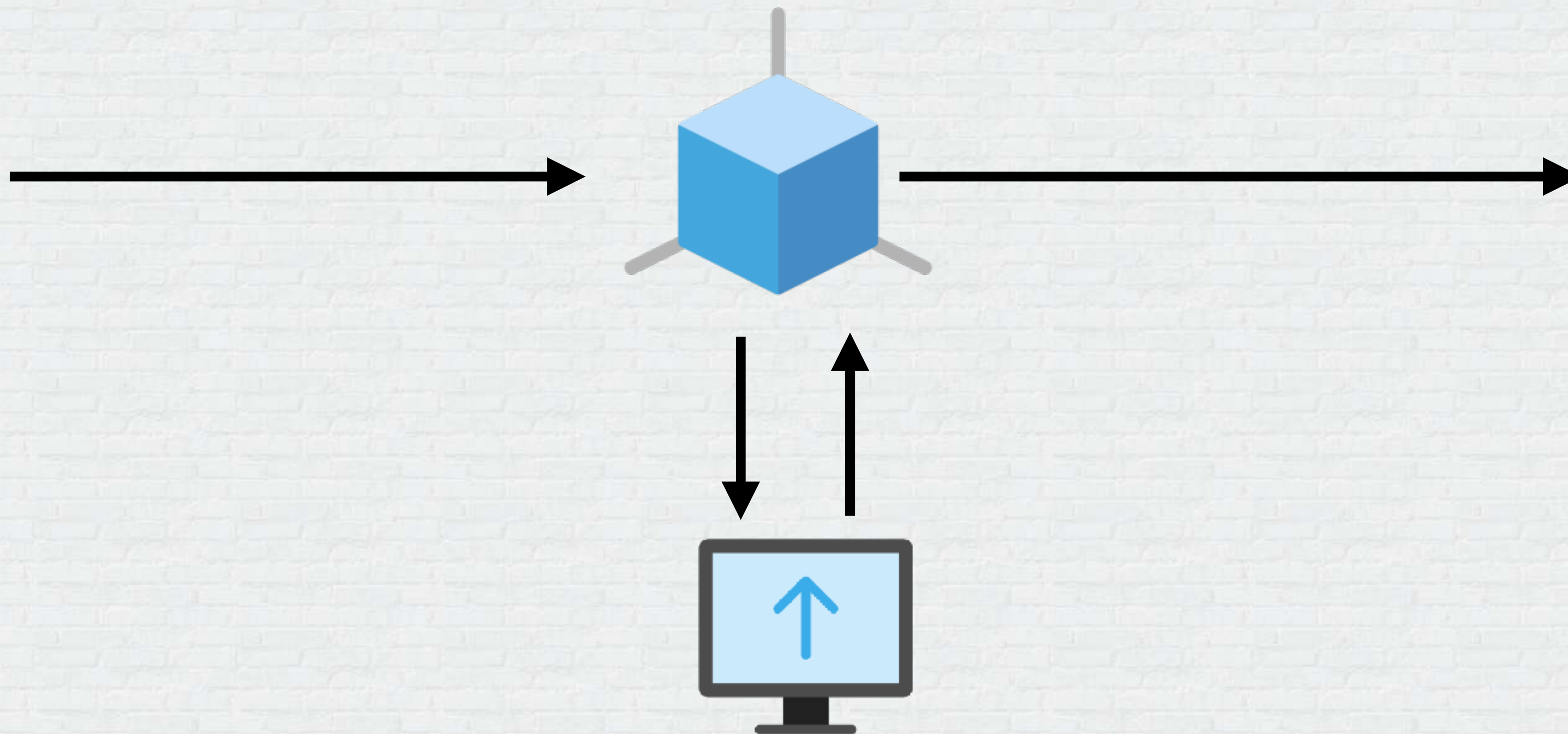


Validating Webhook



# Validating Web Hook

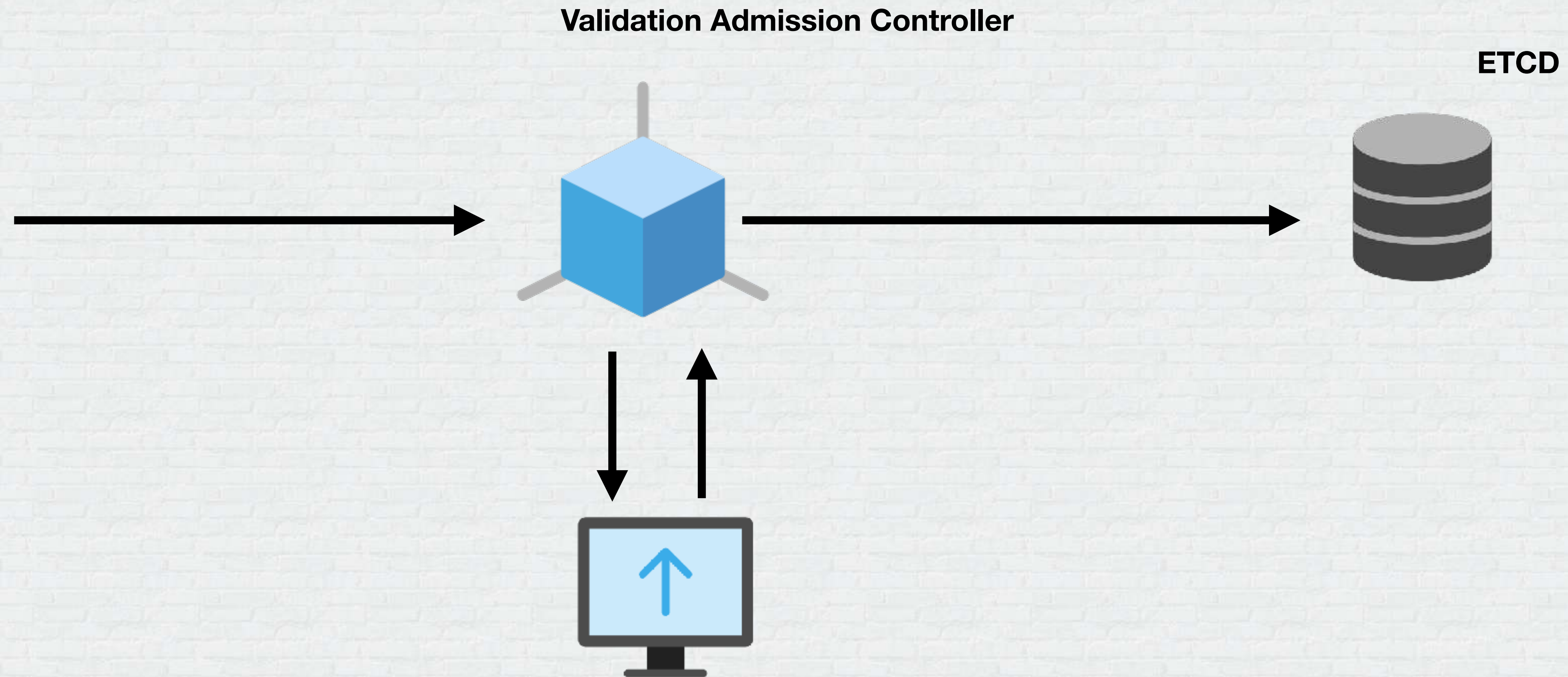
Validation Admission Controller



Validating Webhook



# Validating Web Hook





# Mutating Web Hook





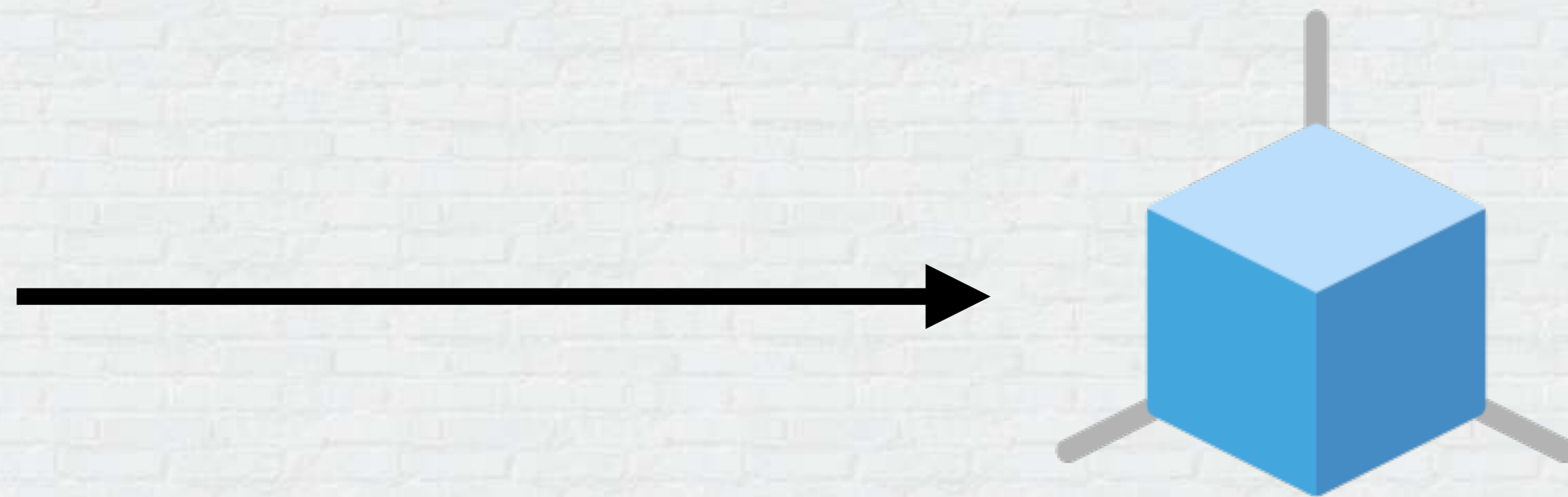
# Mutating Web Hook





# Mutating Web Hook

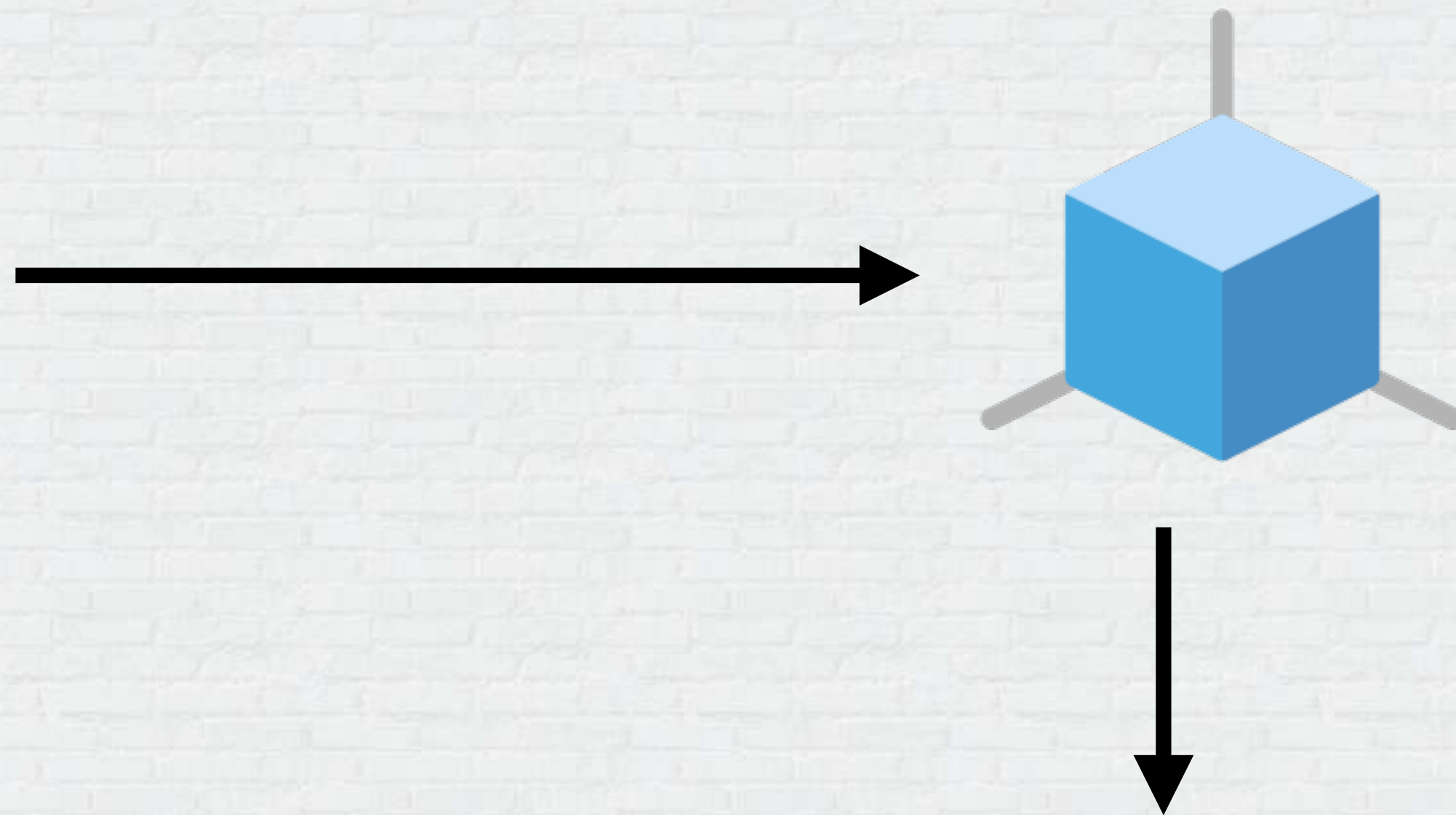
Mutating Admission Controller





# Mutating Web Hook

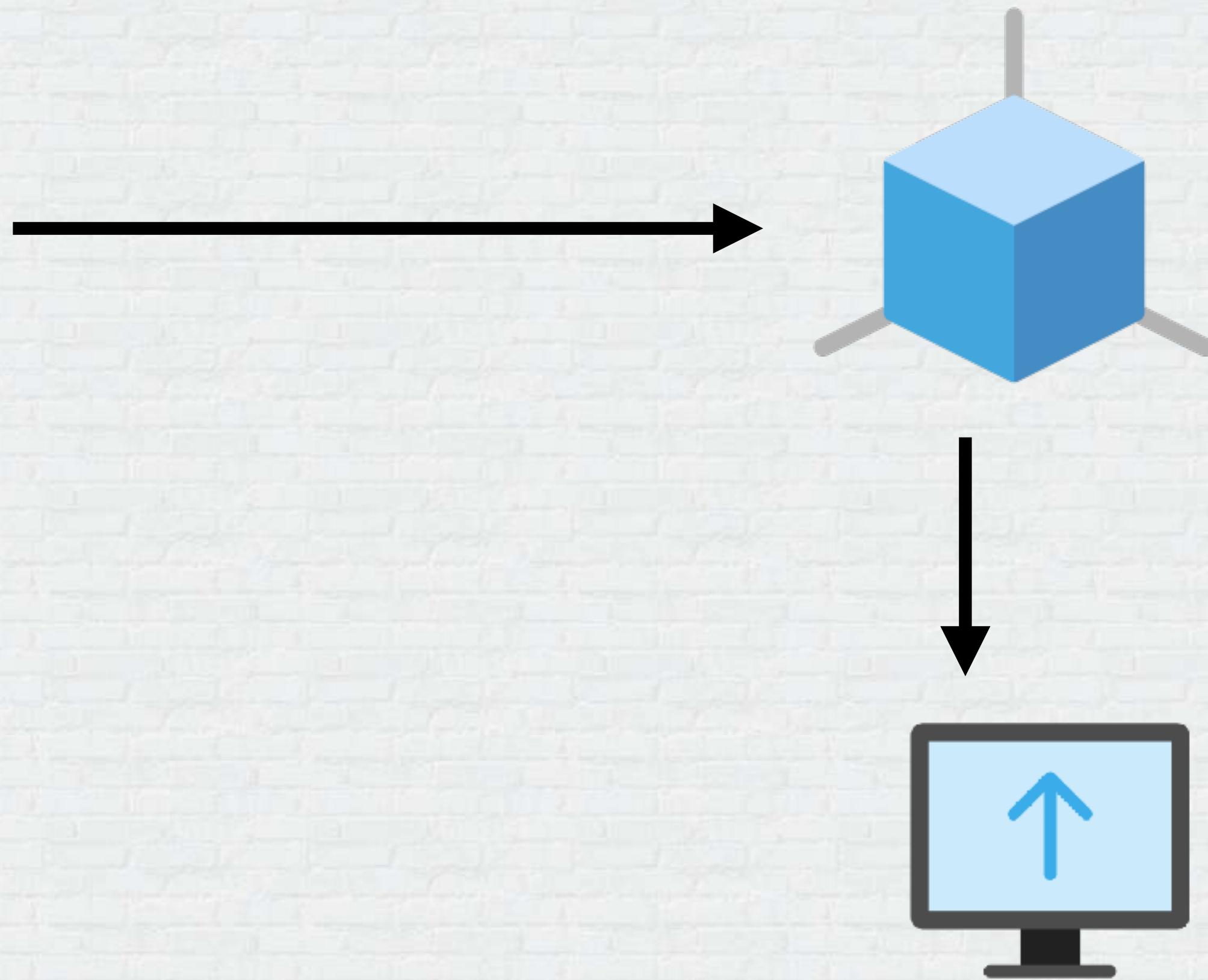
Mutating Admission Controller





# Mutating Web Hook

Mutating Admission Controller



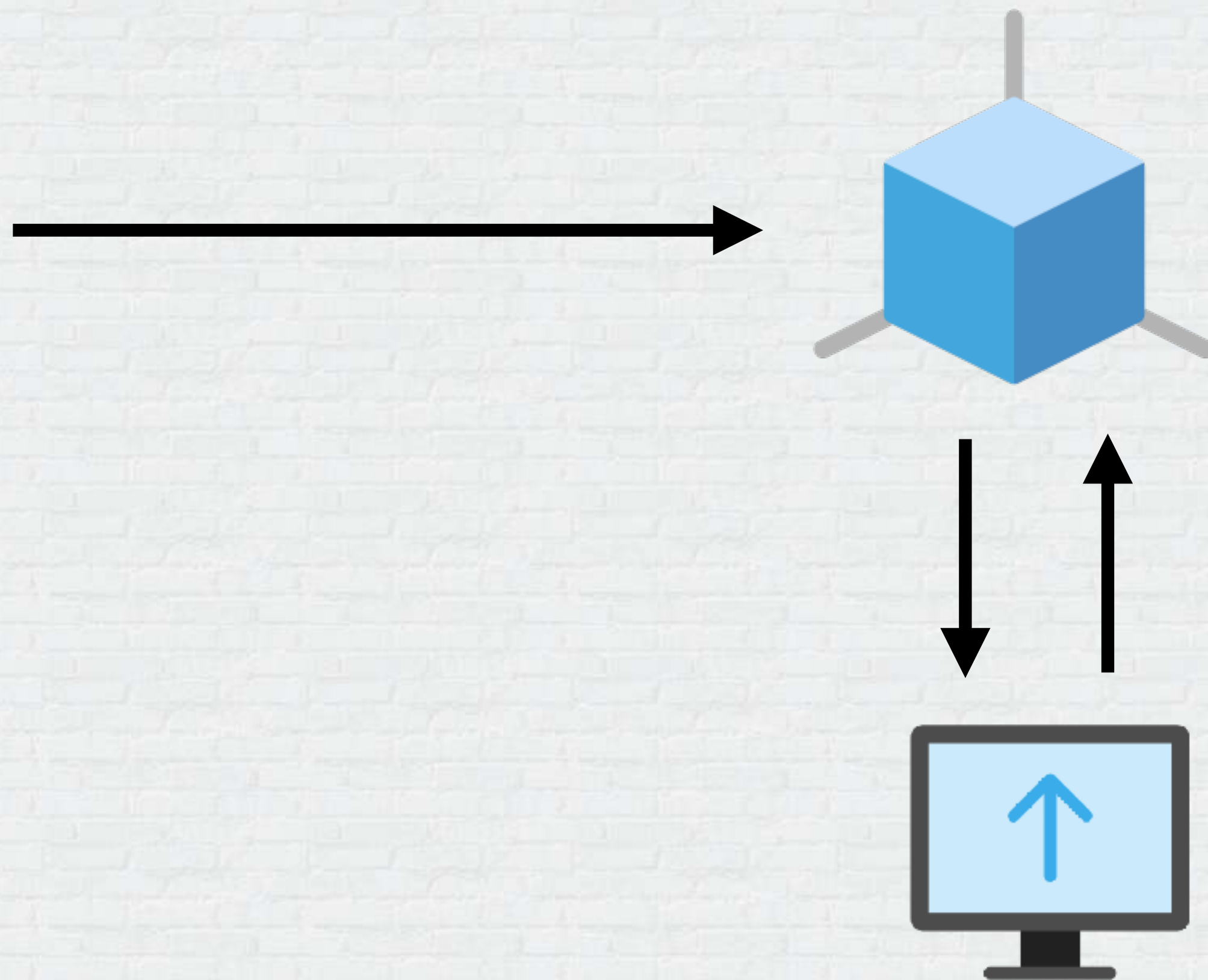
Mutating Webhook

Copyright © AppSecEngineer 2022



# Mutating Web Hook

Mutating Admission Controller



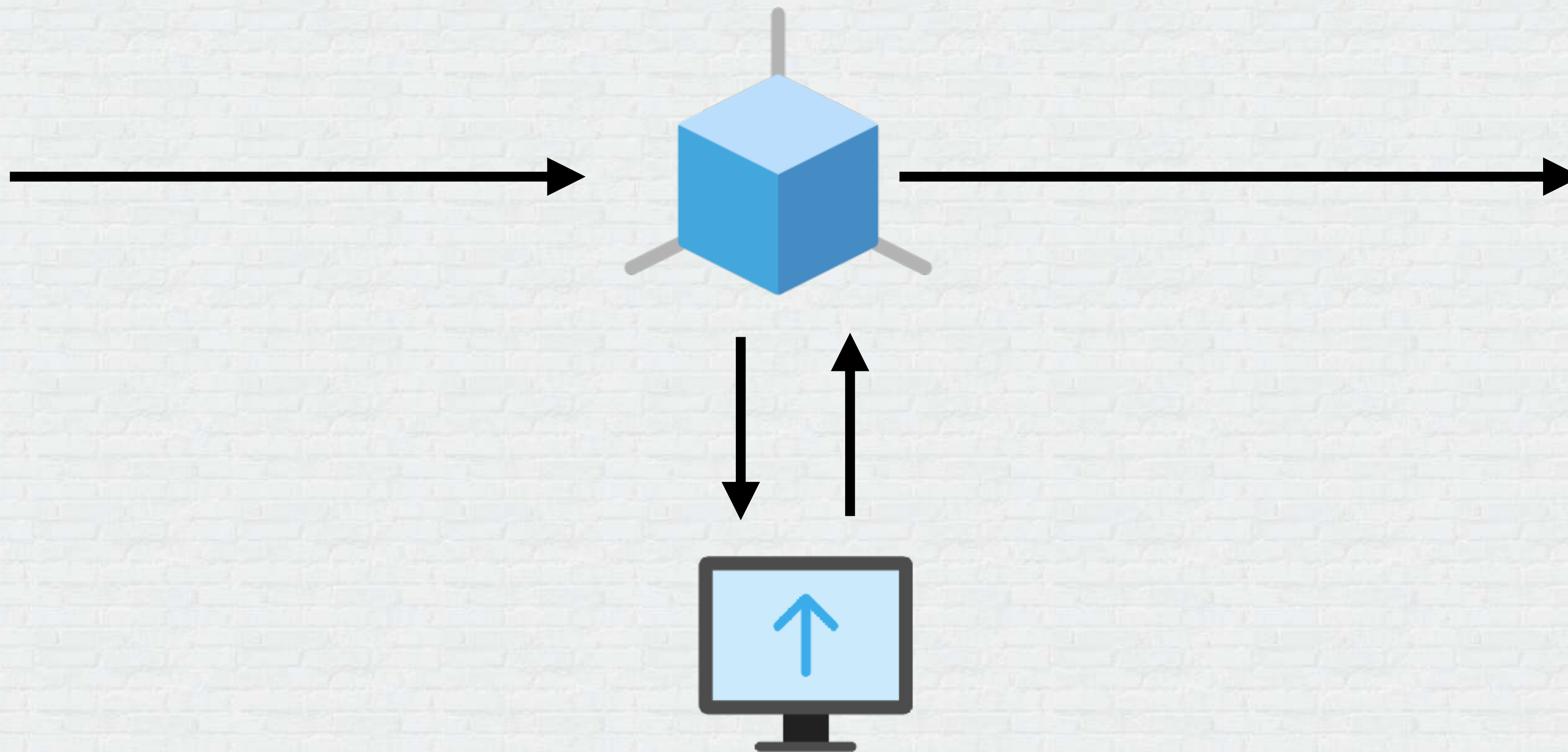
Mutating Webhook

Copyright © AppSecEngineer 2022



# Mutating Web Hook

Mutating Admission Controller

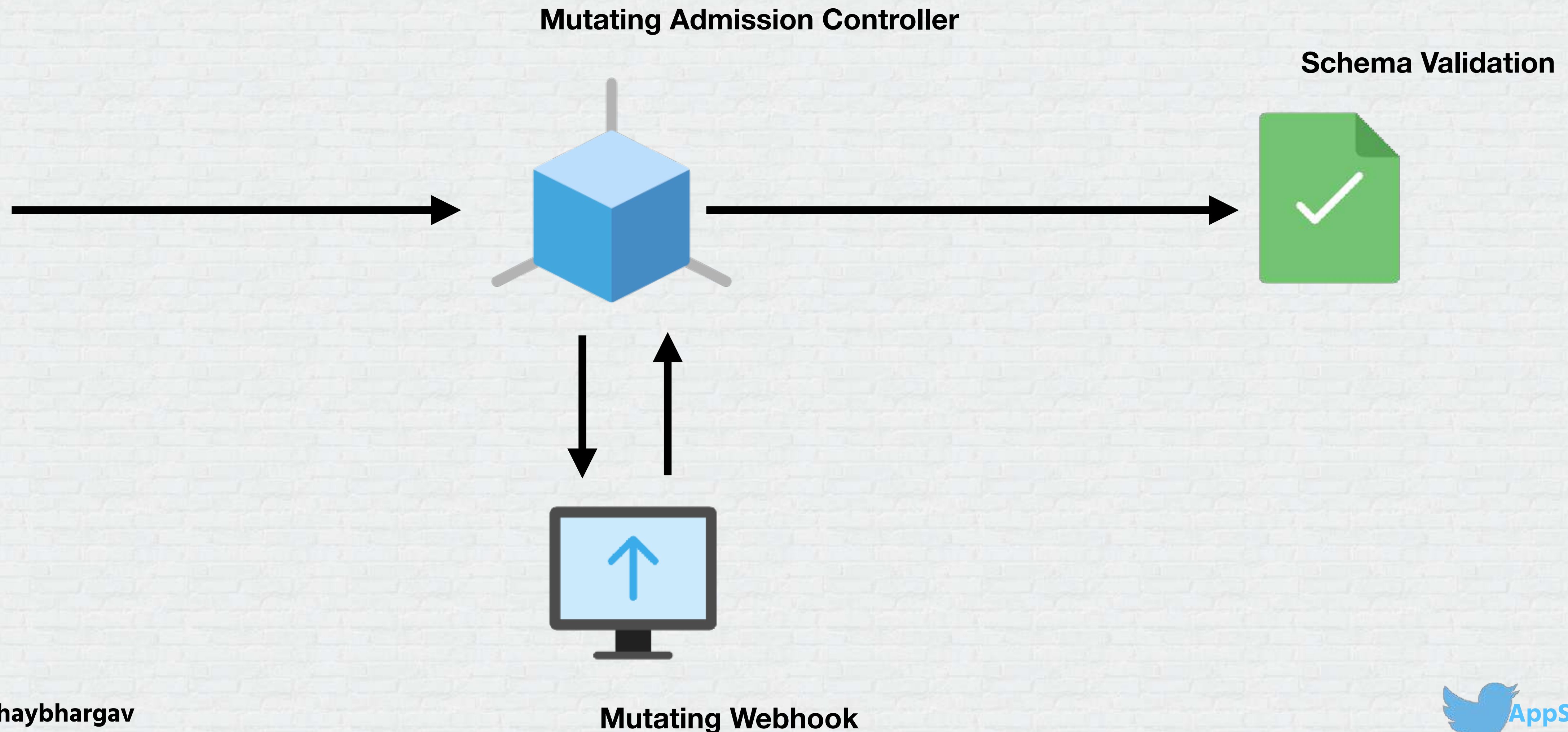


Mutating Webhook

Copyright © AppSecEngineer 2022



# Mutating Web Hook





# Registering the Admission Controller

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration/MutatingWebhookConfiguration
metadata:
  name: "pod-policy.appsecengineer.com"
webhooks:
- name: "pod-policy.appsecengineer.com"
  rules:
  - apiGroups:  [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "webhook-namespace"
      name: "webhook-service"
      caBundle: "CA Bundle to validate the server Certificate"
  admissionReviewVersions: ["v1", "v1beta1"]
  timeoutSeconds: 5
```



# Validating Webhook Response





# Validating Webhook Response

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": true  
  }  
}
```

Allowed Response



# Validating Webhook Response

```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "response": {
    "uid": "<value from request.uid>",
    "allowed": true
  }
}
```

Allowed Response

```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "response": {
    "uid": "<value from request.uid>",
    "allowed": false
  }
}
```

Denied Response



# Validating Webhook Response

```
{  
  "apiVersion":  
  "kind": "Admis  
  "response": {  
    "uid": "<va  
    "allowed": t  
  }  
}
```

Allowed

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": false,  
    "status": {  
      "code": 403,  
      "message": "This request doesn't contain the valid label"  
    }  
  }  
}
```

Denied Response with Custom Messages

```
"admission.k8s.io/v1",  
"AdmissionReview",  
"<value from request.uid>",  
false
```

Denied Response

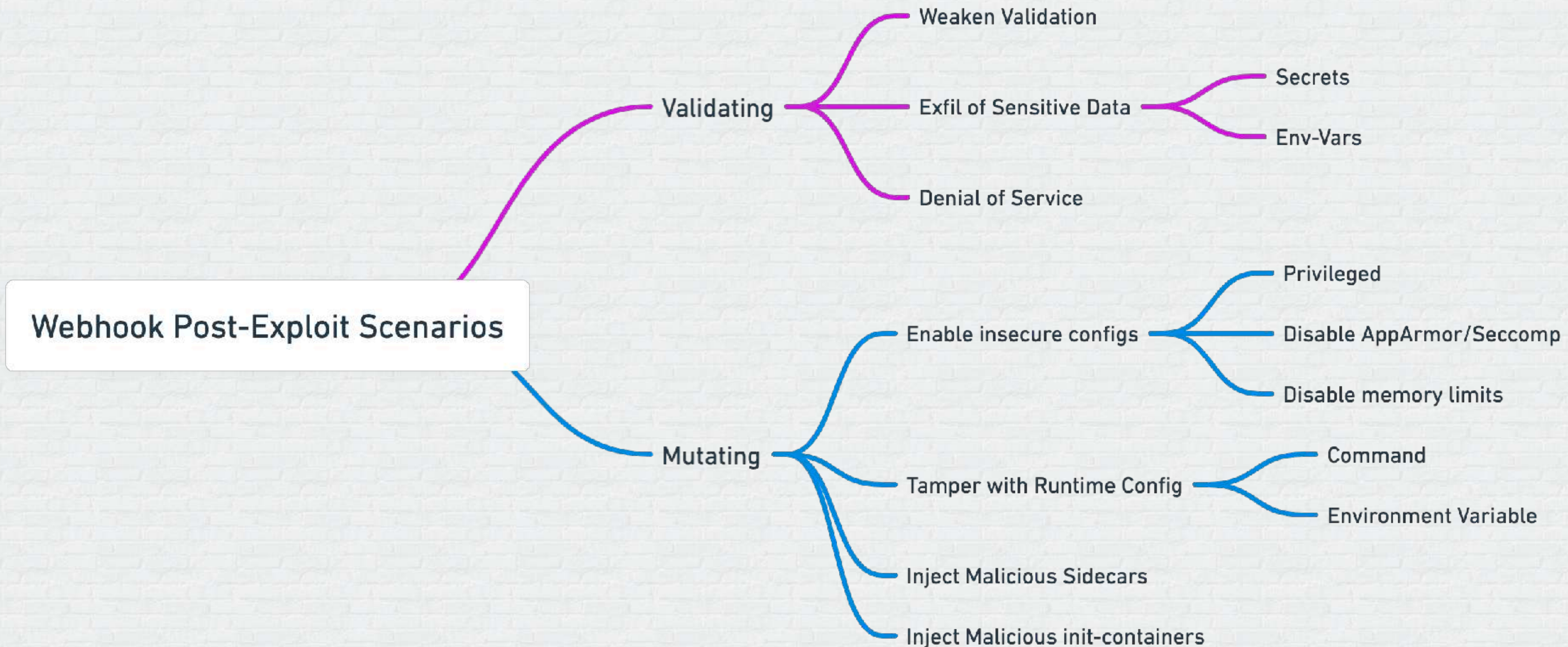


# Response in Mutating Webhook

```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "response": {
    "uid": "<value from request.uid>",
    "allowed": true,
    "patchType": "JSONPatch",
    "patch": "W3sib3AiOiAiYWRkIiwgInBhdGgiOiAiL3NwZWVibGFiZWwiLCBidmFsdWUiOiAiYXBwc2VjZW5naW5lZXIifV0="
  }
}
```



# Possible Post-Exploit Scenarios





# Demo

