Image Search with Kafka, S3, Elastic, and LLMs

Giulia Cuppoletti, Alec Davis, Isabella Staley

HPCCEA 2025

Prepared by LLNL under Contract DE-AC52-07NA27344.

Meet the team

Giulia Cuppoletti
Major: Computer Science
School: Penn State
Year: Recently graduated

Alec Davis
Major: Computer Science
School: BYU
Year: Senior

Isabella Staley
Major: Information Technology
School: Wayne State University
Year: Junior

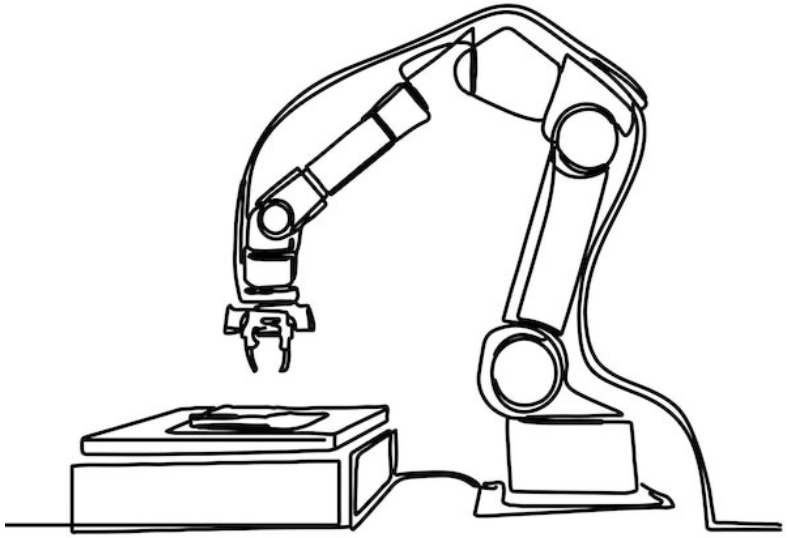
What did we want to build?

An image search engine that could:

- Search images based on key features
- Quickly yield search results from many images
- Have a pipeline that could be spread across multiple compute nodes/clusters

Why did we want to build it?

- Proof of concept for clients using automated lab equipment
  - Many pictures taken autonomously
  - Need for analysis/categorization of pictures
  - Wanted ability to search through pictures
- Explore interaction of Kafka, Elastic, and S3 on clusters



Technologies We Used

Apache Kafka

- Data streaming
- Consumers and Producers
- Can be distributed across nodes/clusters

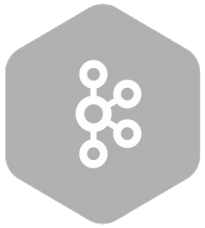
NetApp Storage Grid

- Imitates S3 (Simple Storage Service)
- Stores objects in buckets
- Objects available through URL

Elasticsearch

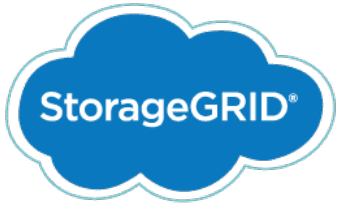
- Powerful search engine
- Stores data in indexes
- Allows for complex queries (keywords, filters, etc.)

Technologies We Used



Apache Kafka

- Data streaming
- Consumers and Producers
- Can be distributed across nodes/clusters



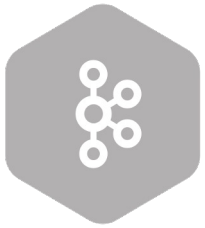
NetApp Storage Grid

- Imitates S3 (Simple Storage Service)
- Stores objects in buckets
- Objects available through URL

Elasticsearch

- Powerful search engine
- Stores data in indexes
- Allows for complex queries (keywords, filters, etc.)

Technologies We Used



Apache Kafka

- Data streaming
- Consumers and Producers
- Can be distributed across nodes/clusters

NetApp Storage Grid

- Imitates S3 (Simple Storage Service)
- Stores objects in buckets
- Objects available through URL

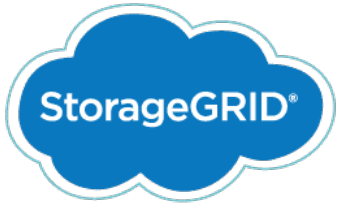
Elasticsearch

- Powerful search engine
- Stores data in indexes
- Allows for complex queries (keywords, filters, etc.)

Technologies We Used

Apache Kafka

- Data streaming
- Consumers and Producers
- Can be distributed across nodes/clusters



NetApp Storage Grid

- Imitates S3 (Simple Storage Service)
- Stores objects in buckets
- Objects available through URL

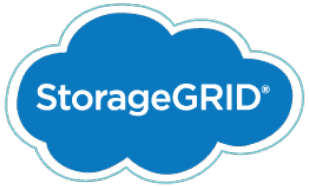
Elasticsearch

- Powerful search engine
- Stores data in indexes
- Allows for complex queries (keywords, filters, etc.)

How We Did It

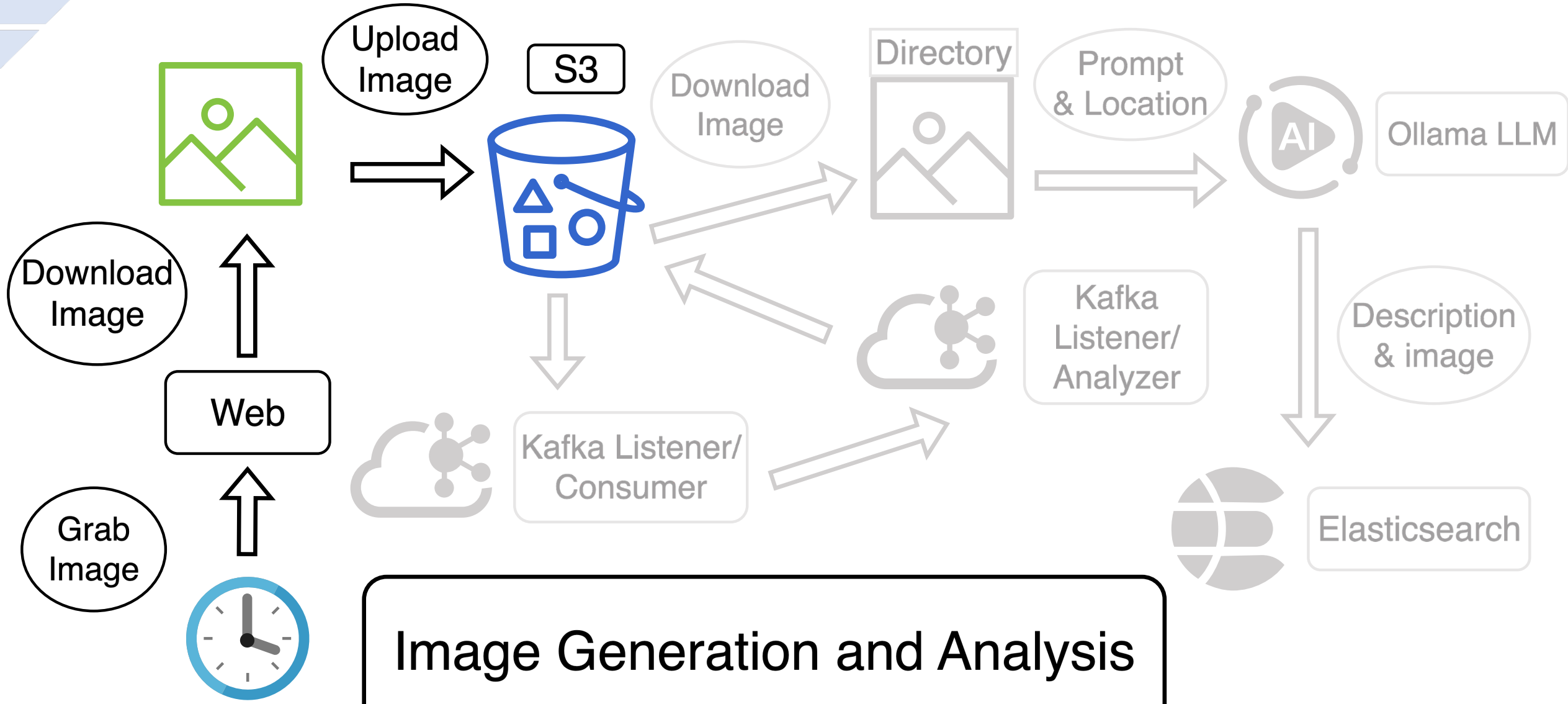
Set Up

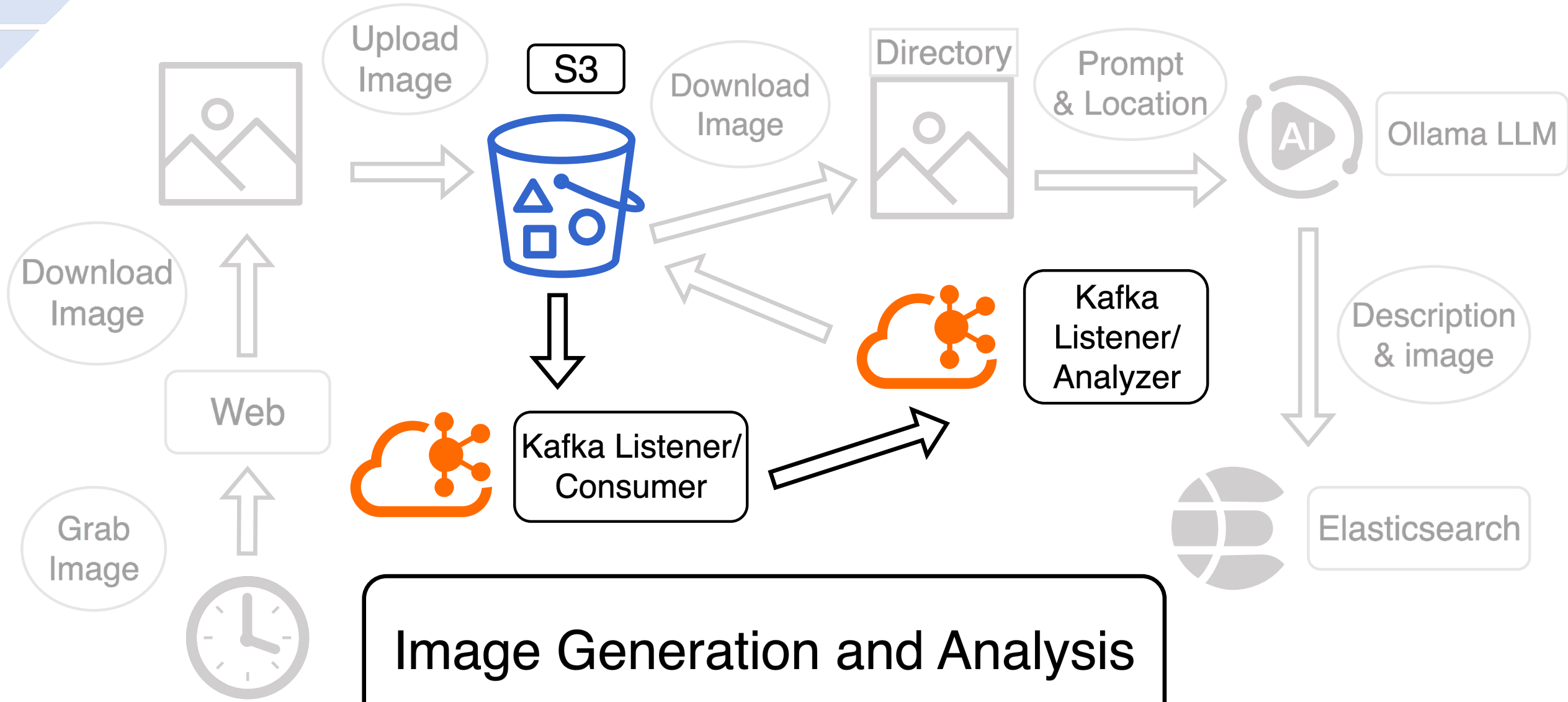
- Kafka Container
- NetApp Storage Grid
- Elasticsearch
- Ollama (OpenShift)
  - Llama 4 Scout

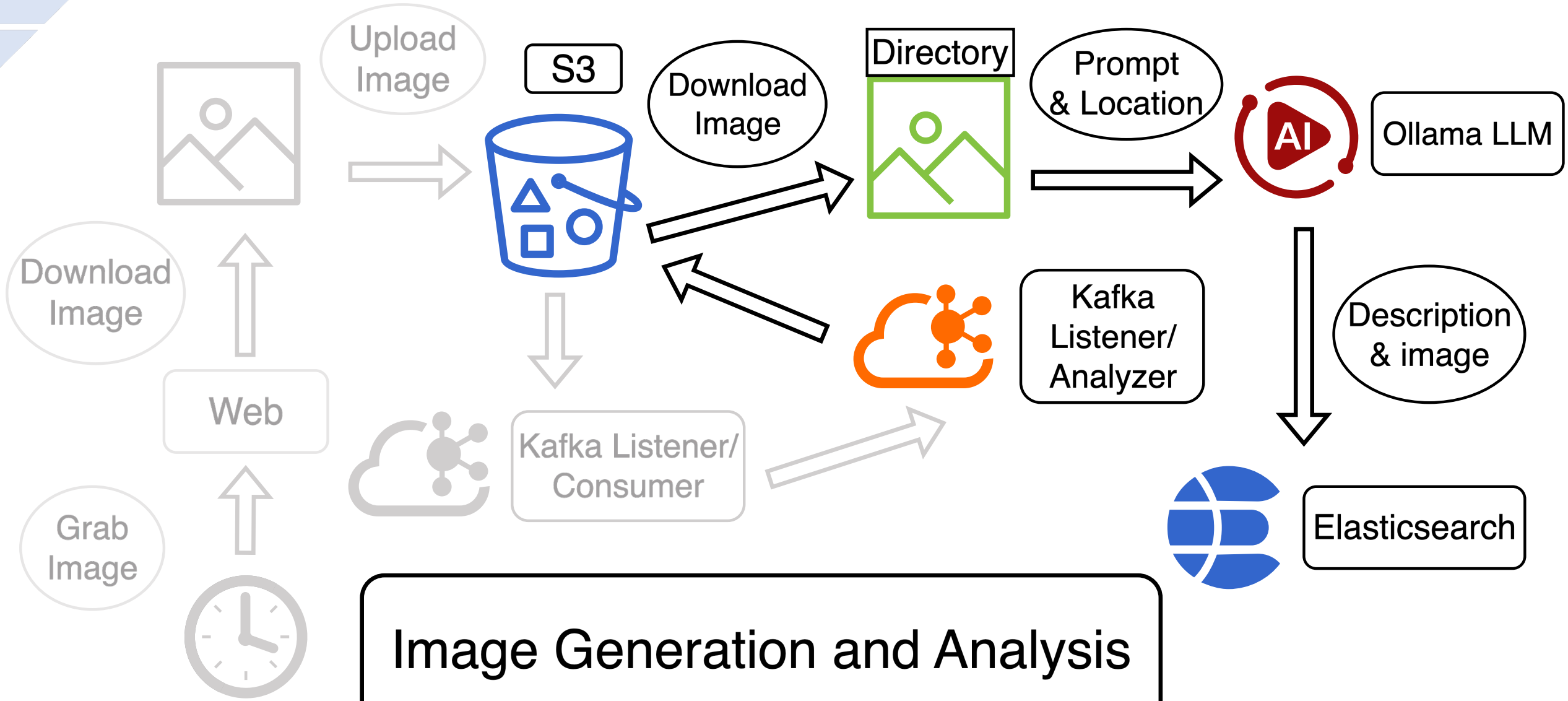


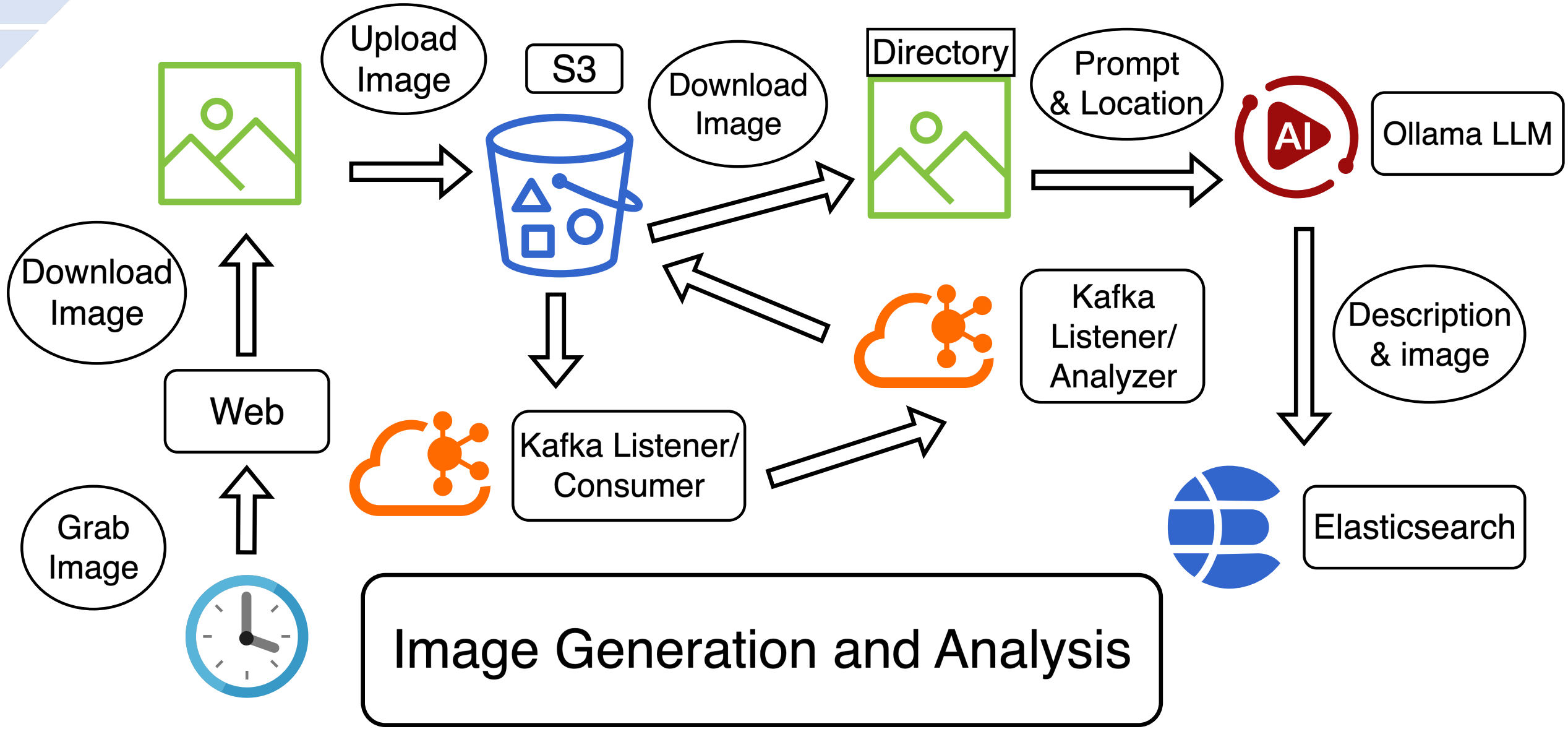
Implementation Workflow

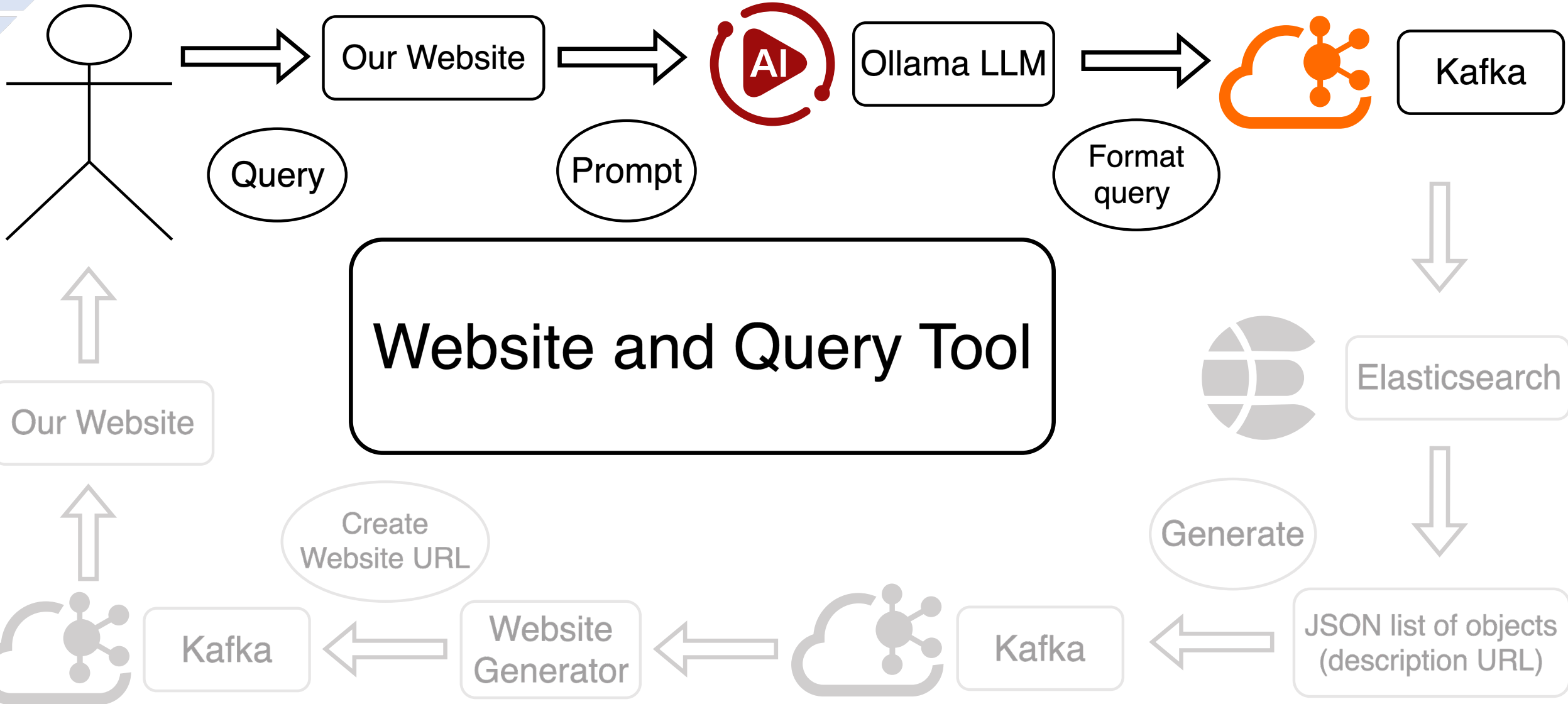
1. Image Generator
2. Image Consumer
3. Image Analyzer
4. Website
5. Query Tool
6. Webpage Generator

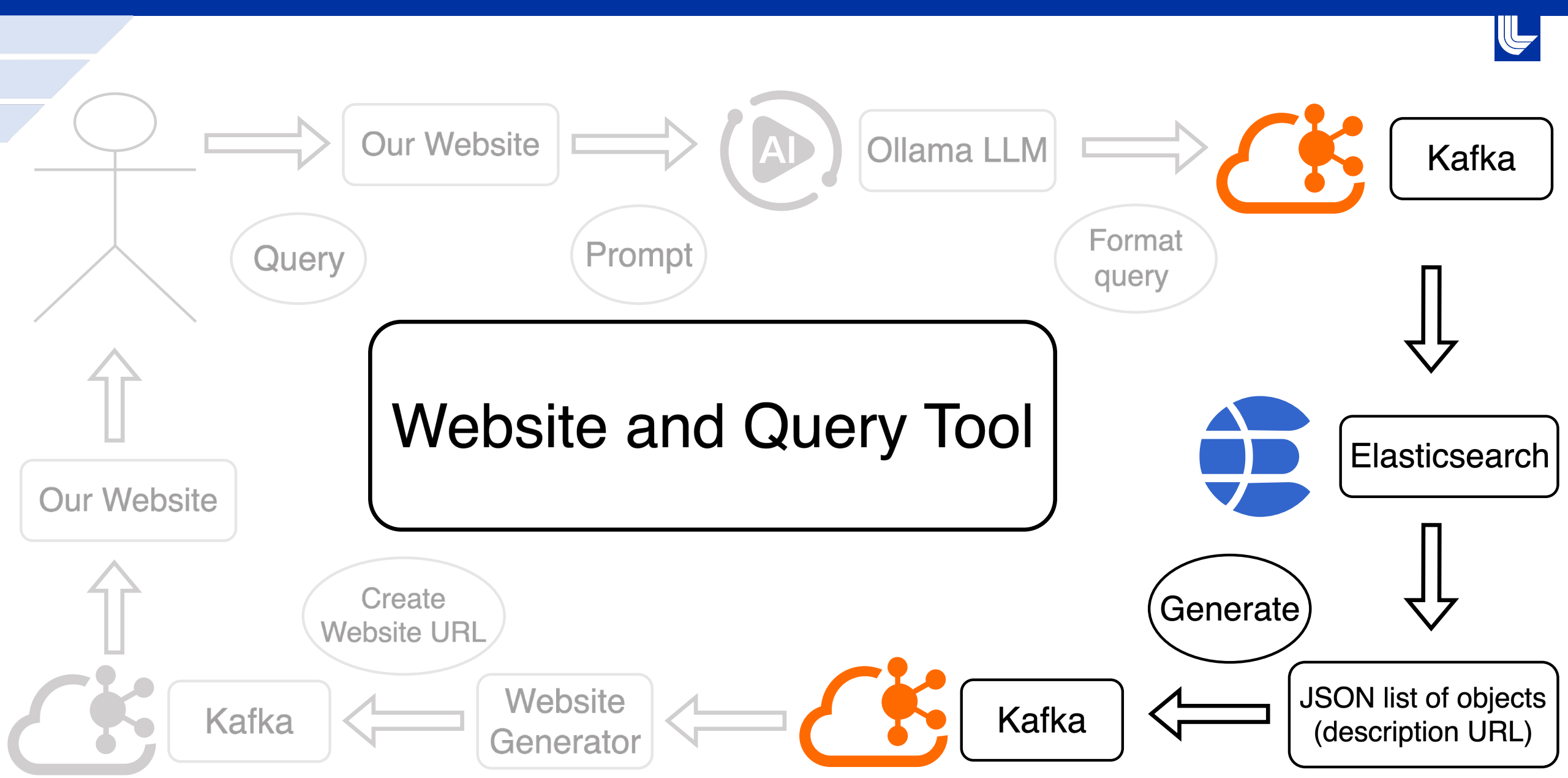

Upload Image
S3
Download Image
Directory
Prompt & Location
AI
Ollama LLM
Download Image
Web
Kafka Listener/ Analyzer
Description & image
Kafka Listener/ Consumer
Elasticsearch
Grab Image
Image Generation and Analysis

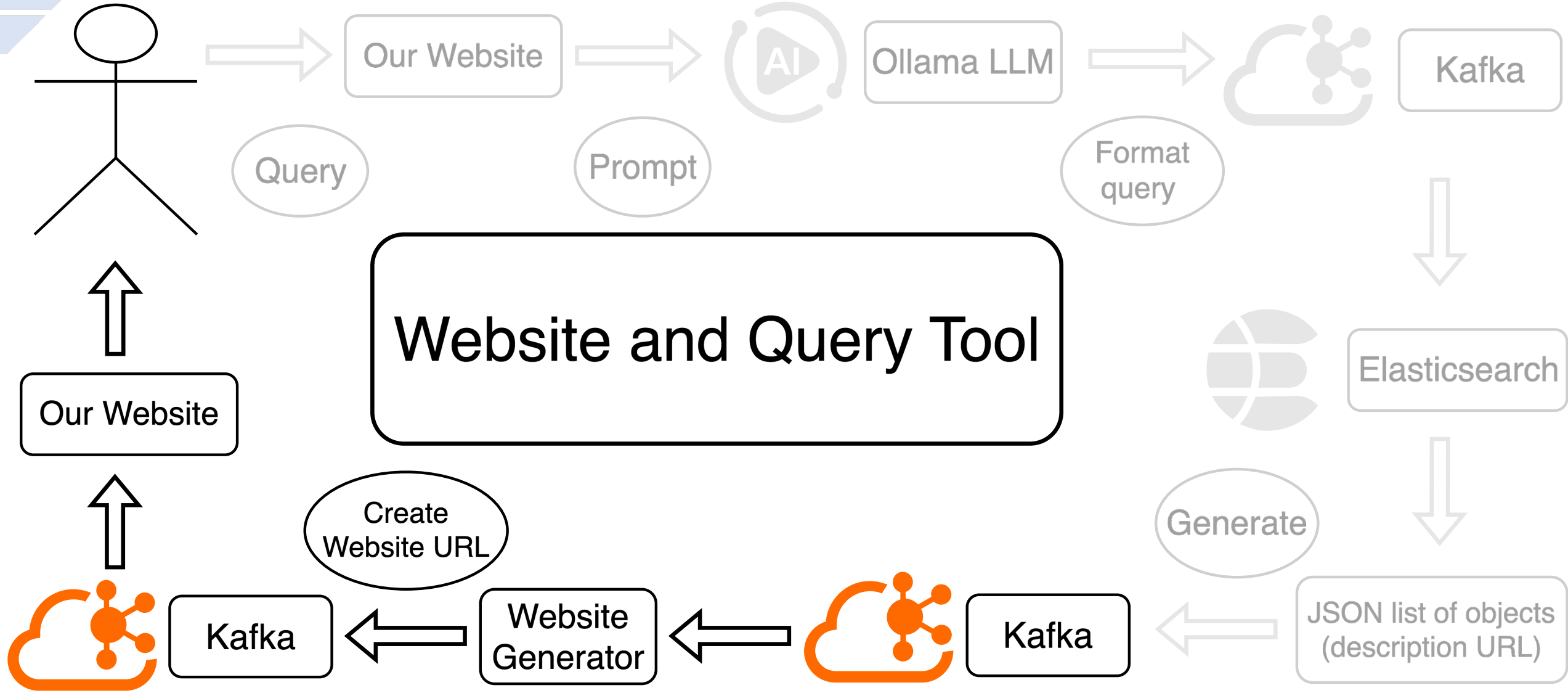

Upload Image
S3
Download Image
Directory
Prompt & Location
AI
Ollama LLM
Download Image
Web
Grab Image
Kafka Listener/ Consumer
Kafka Listener/ Analyzer
Description & image
Elasticsearch
Image Generation and Analysis

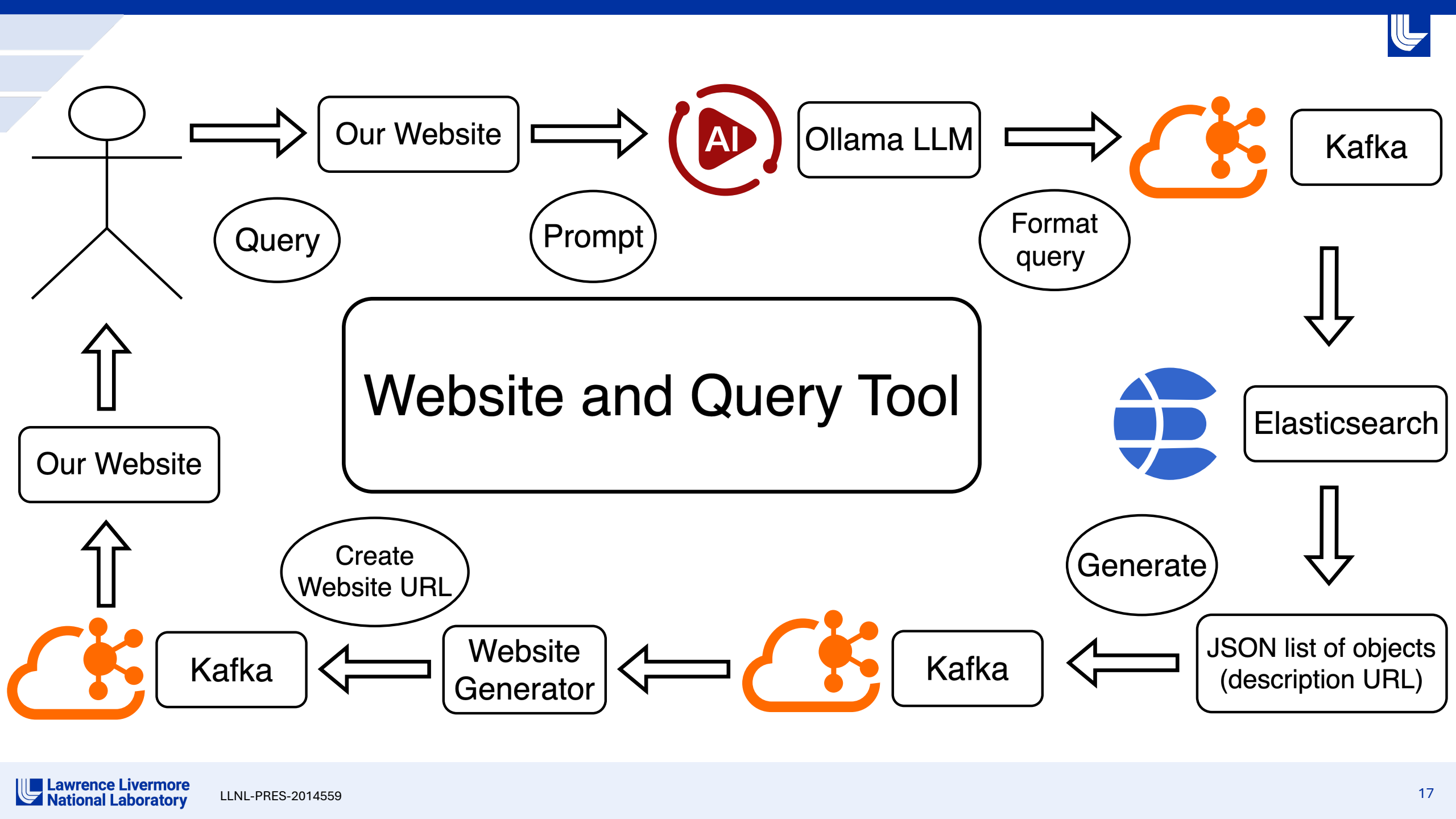

Upload Image
S3
Download Image
Directory
Prompt & Location
Ollama LLM
Download Image
Web
Kafka Listener/ Consumer
Kafka Listener/ Analyzer
Description & image
Grab Image
Elasticsearch
Image Generation and Analysis


Upload Image
S3
Download Image
Directory
Prompt & Location
AI
Ollama LLM
Download Image
Web
Kafka Listener/ Analyzer
Description & image
Kafka Listener/ Consumer
Elasticsearch
Grab Image
Image Generation and Analysis


Our Website
Ollama LLM
Kafka
Query
Prompt
Format query
Website and Query Tool
Our Website
Elasticsearch
Create Website URL
Generate
Kafka
Website Generator
Kafka
JSON list of objects (description URL)


Our Website
Ollama LLM
Kafka
Query
Prompt
Format query
Website and Query Tool
Our Website
Elasticsearch
Generate
Create Website URL
Kafka
Website Generator
Kafka
JSON list of objects (description URL)


Our Website
Ollama LLM
Kafka
Query
Prompt
Format query
Website and Query Tool
Our Website
Elasticsearch
Generate
Create Website URL
Kafka
Website Generator
Kafka
JSON list of objects (description URL)


Our Website
Ollama LLM
Kafka
Query
Prompt
Format query
Website and Query Tool
Our Website
Elasticsearch
Create Website URL
Generate
Kafka
Website Generator
Kafka
JSON list of objects (description URL)

Demo

Image Generation Pipeline

Step 1 Create Image:

- We pull an image from Unsplash
- Put it in Storagegrid (S3)
- Storagegrid sends out a Kafka Notification

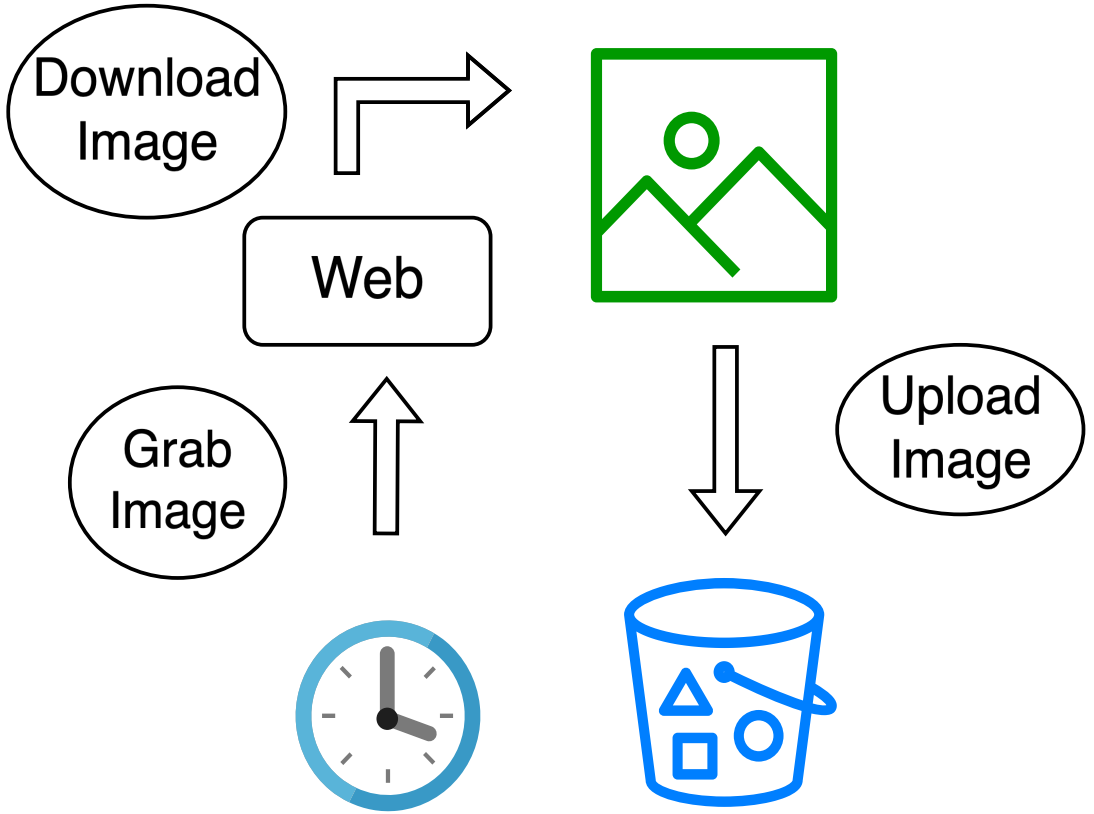


Image Generation Pipeline

Step 2 Notify Analyzers:

- Respond to Kafka Notification
- Notify Analyzer via Kafka and provide the URL to the image in Storagegrid

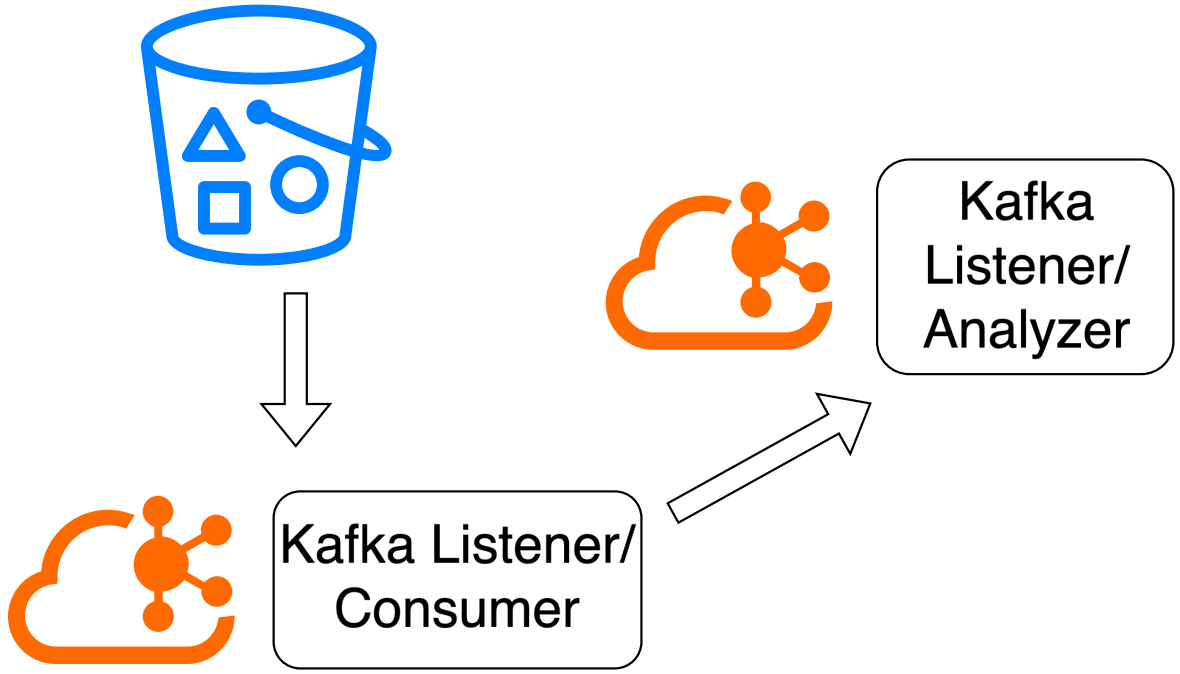


Image Generation Pipeline

Step 3 Analyze Image:

- Respond to Kafka Notification
- Download image
- Prompt LLM via Ollama for Natural Language Description
- Prompt LLM again with NL Desc to get Keyword Desc
- Put URL, NL Desc, and Keyword Desc in Elasticsearch

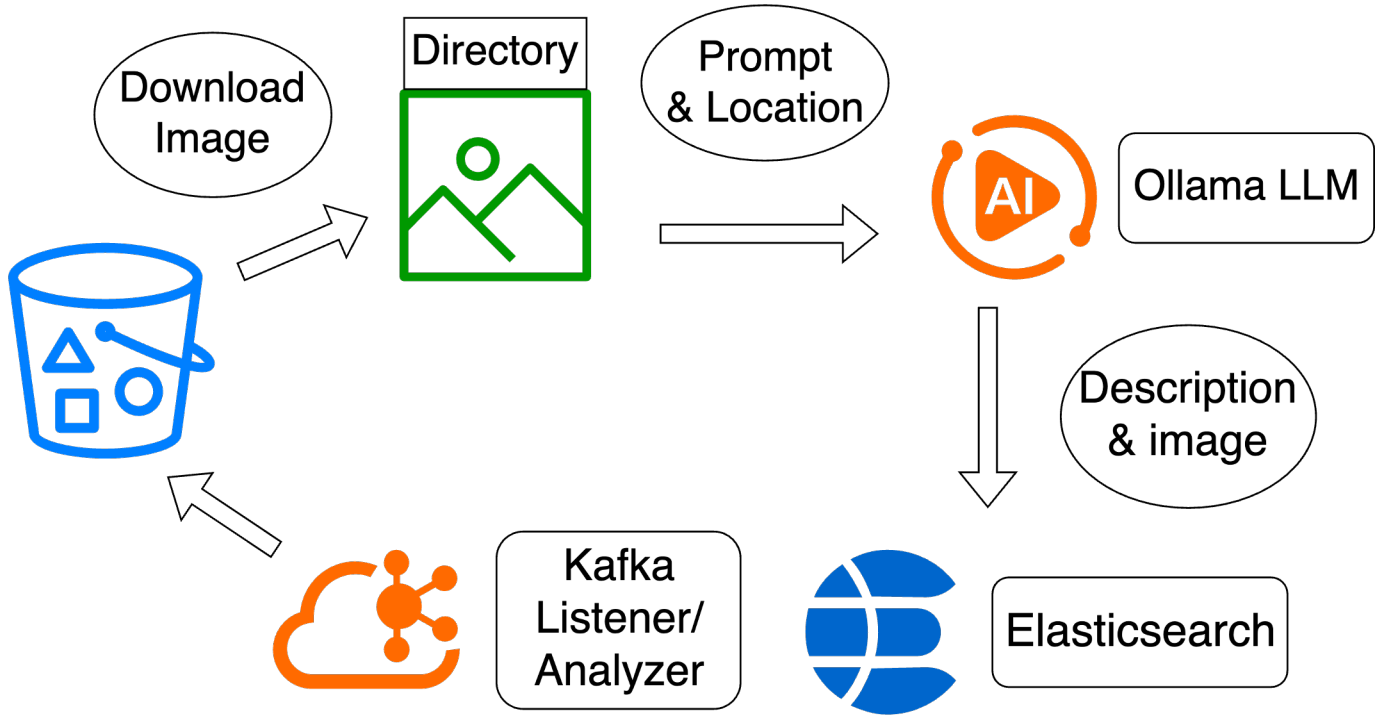


Image Query Pipeline

Step 1 User Website:

- User uses a (somewhat) natural language query
- Backend uses Ollama to turn query into keywords
- Posts both query and keywords to Kafka for Query Tool

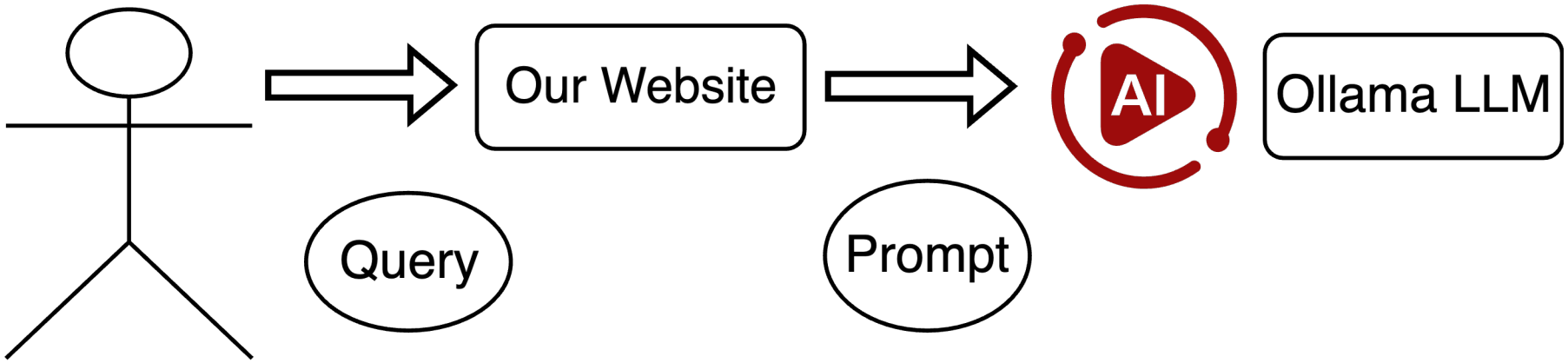


Image Query Pipeline

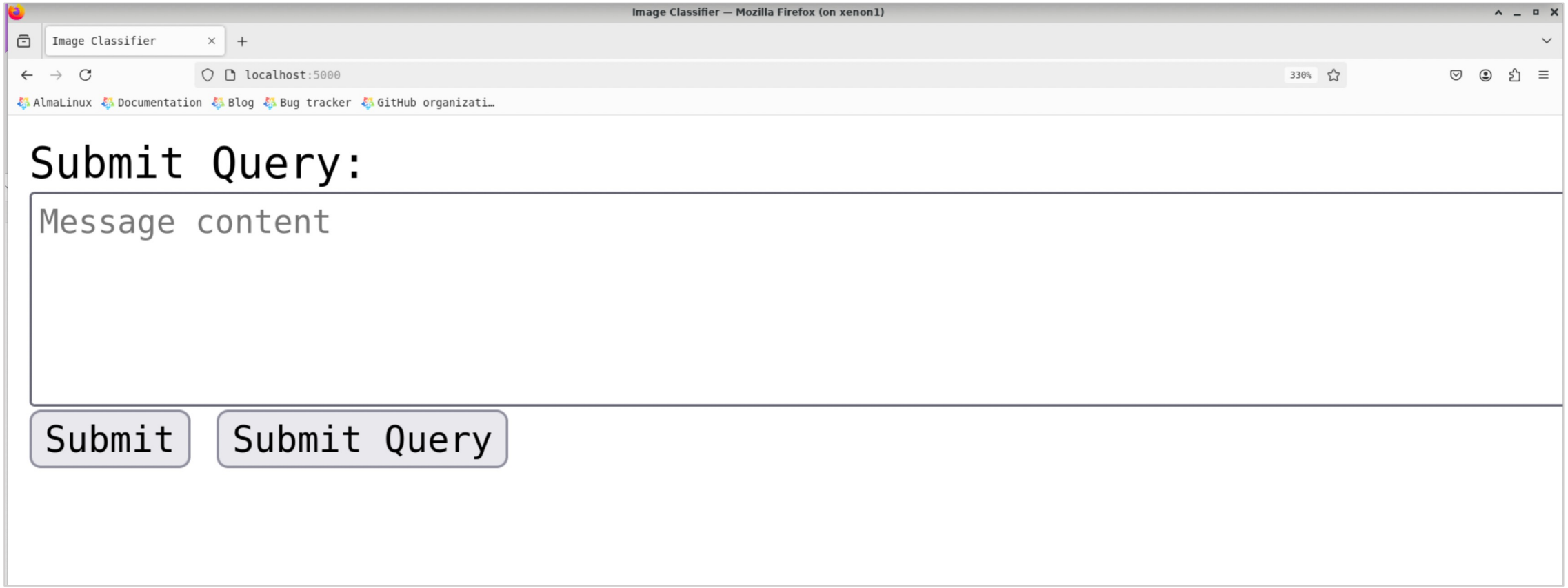


Image Query Pipeline

Step 2 Query Tool:

- Responds to Kafka Message
- Uses Keywords to find images that match all keywords in Elasticsearch
- Posts results and the query to Webpage Generator

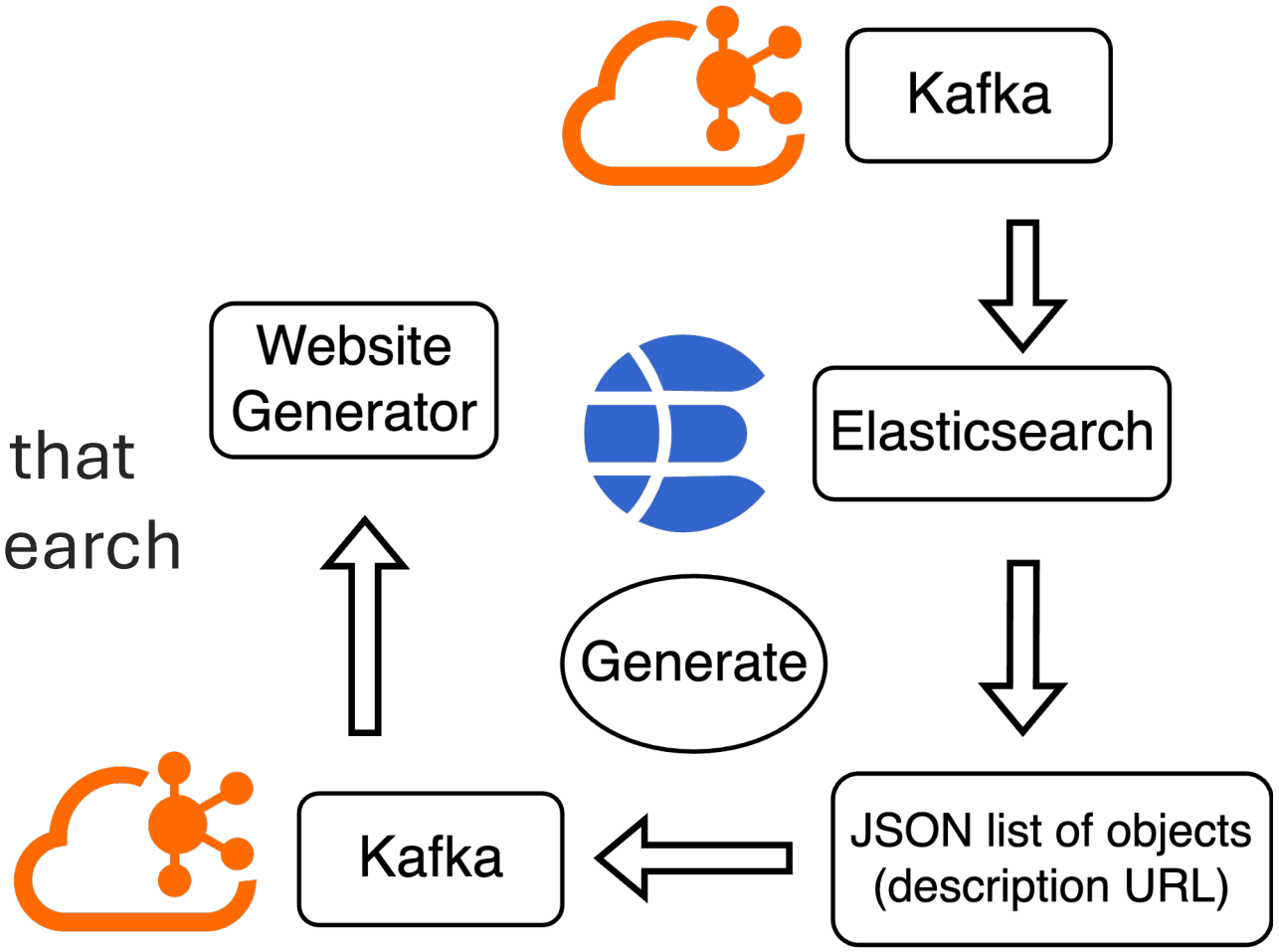


Image Query Pipeline

Step 3 Webpage Generator:

- Responds to Kafka Message
- Builds a static webpage to show the results of the user's query
- Posts a message back to website with the URL of the newly generated webpage

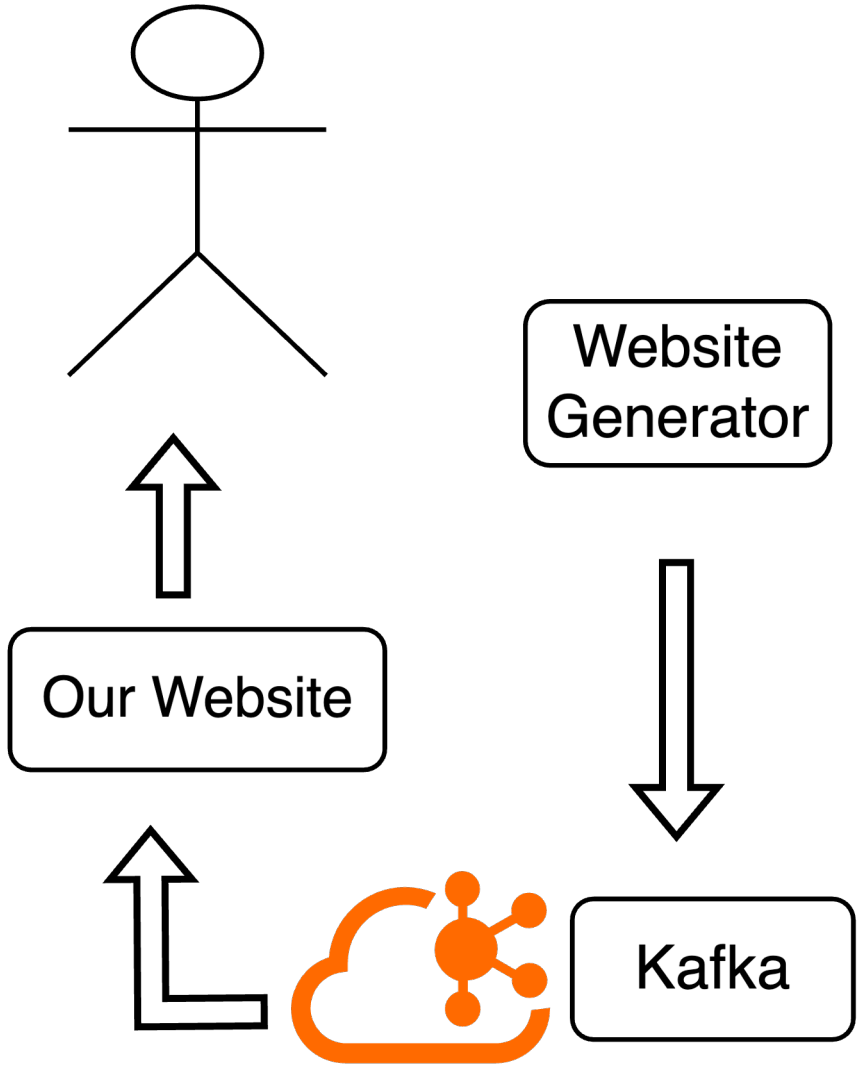


Image Query Pipeline

Step 4 Redirect User:

- Responds to Kafka Message
- Redirect user to newly created webpage to show the results.

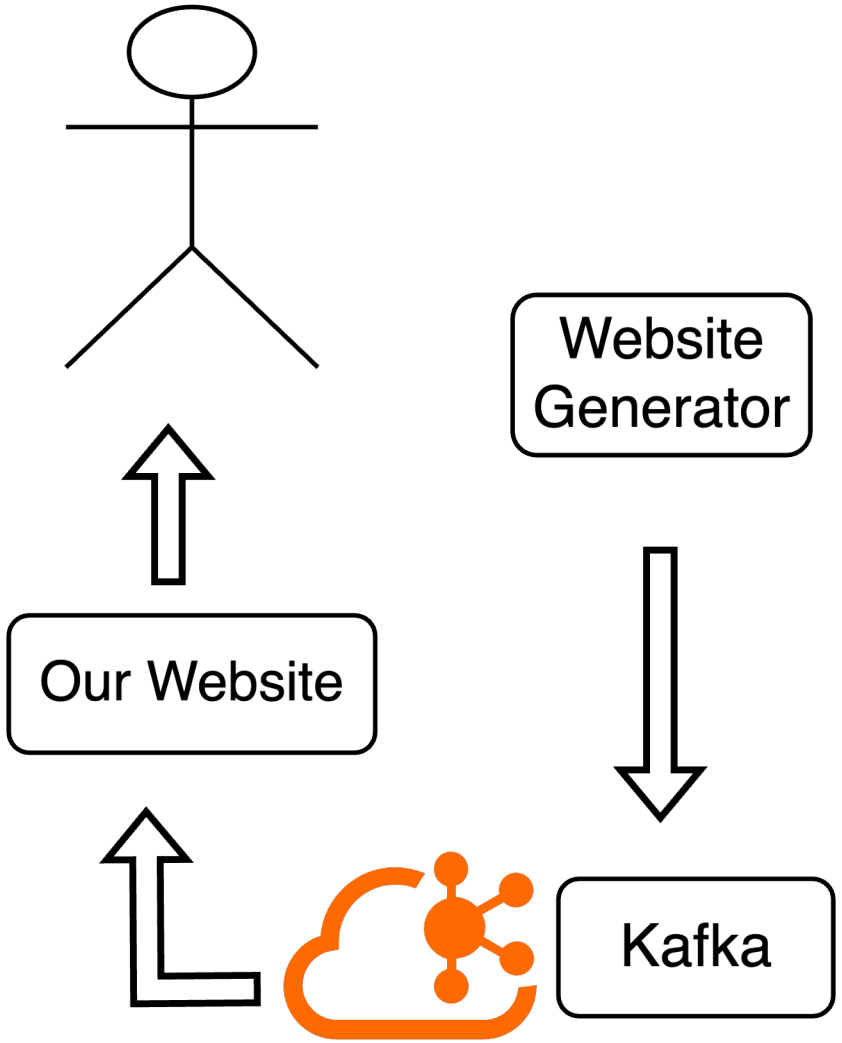
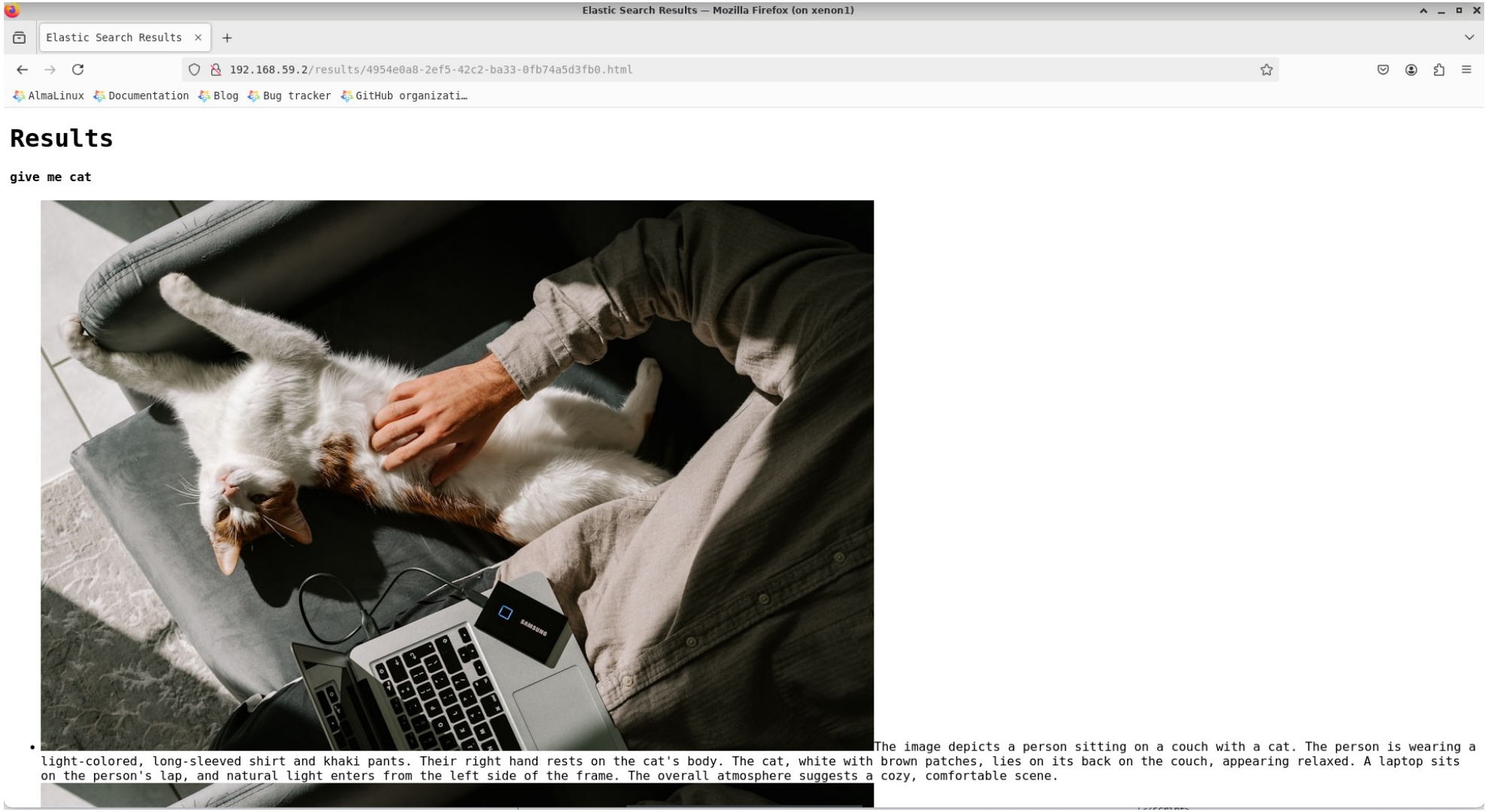


Image Query Pipeline



Automation…because running 6 python scripts by hand every time is a pain!

- Ansible playbook deployment
- Systemd services
- Centralized config files

Challenges

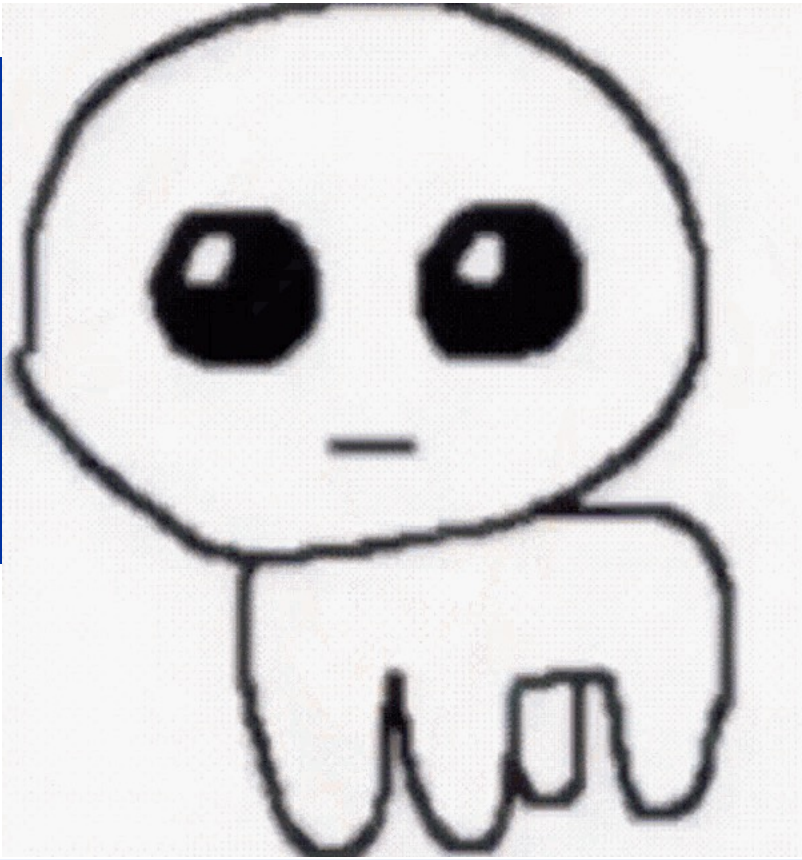
- Learning and using the required technologies
  - Containerizing Kafka
  - Configuring StorageGrid bucket for URL access
- Connecting each step of the pipelined workflow
  - Standardizing input and output
- Engineering LLM prompts

Next Steps

- Fine tune the LLM prompts
- Adjust our tool for lab settings
  - Connect image generation to actual lab equipment
  - Train an LLM on lab pictures
- Test the accuracy and speed of search results with a large sample of pictures

Thank you!

Thank you!



Thank you to our mentors!

Dave Fox

Gabe Maxfield

References

- https://kafka.apache.org/
- https://www.elastic.co/elasticsearch
- https://www.netapp.com/storagegrid/
- https://unsplash.com/
- https://www.freepik.com/
- https://stock.adobe.com/images/
- https://pixabay.com/photos/search/

OpenCHAMI

Open Composable Heterogeneous Adaptable Management Interface

Jasmine Chao and Matthew Torno

Prepared by LLNL under Contract DE-AC52-07NA27344.

Jasmine Chao

Carnegie Mellon University
Electrical and Computer Engineering

Matthew Torno

University of San Diego
Computer Science / Cybersecurity

Prepared by LLNL under Contract DE-AC52-07NA27344.

What is OpenCHAMI?

> “OpenCHAMI … is an open-source system management platform designed to bring *cloud-like flexibility and security* to HPC environments.

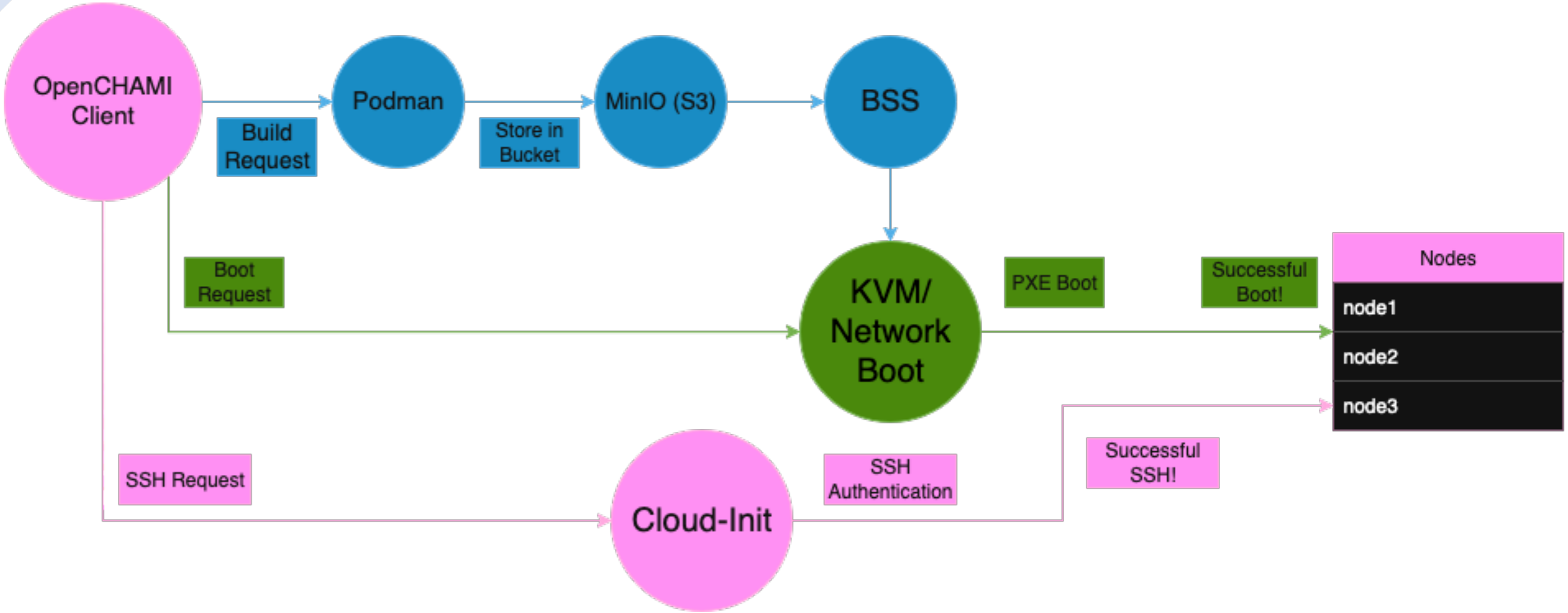
(from the OpenCHAMI official website at https://openchami.org/)

Features

- Makes resource provisioning **simple and modular**
  - Images can be built in any way using a **layer system**
  - Once the images are built, they can be booted any number of times in about a minute
- Features **automated authentication**
  - Integrated cloud-init
  - Manages SSH access, boot parameters and more

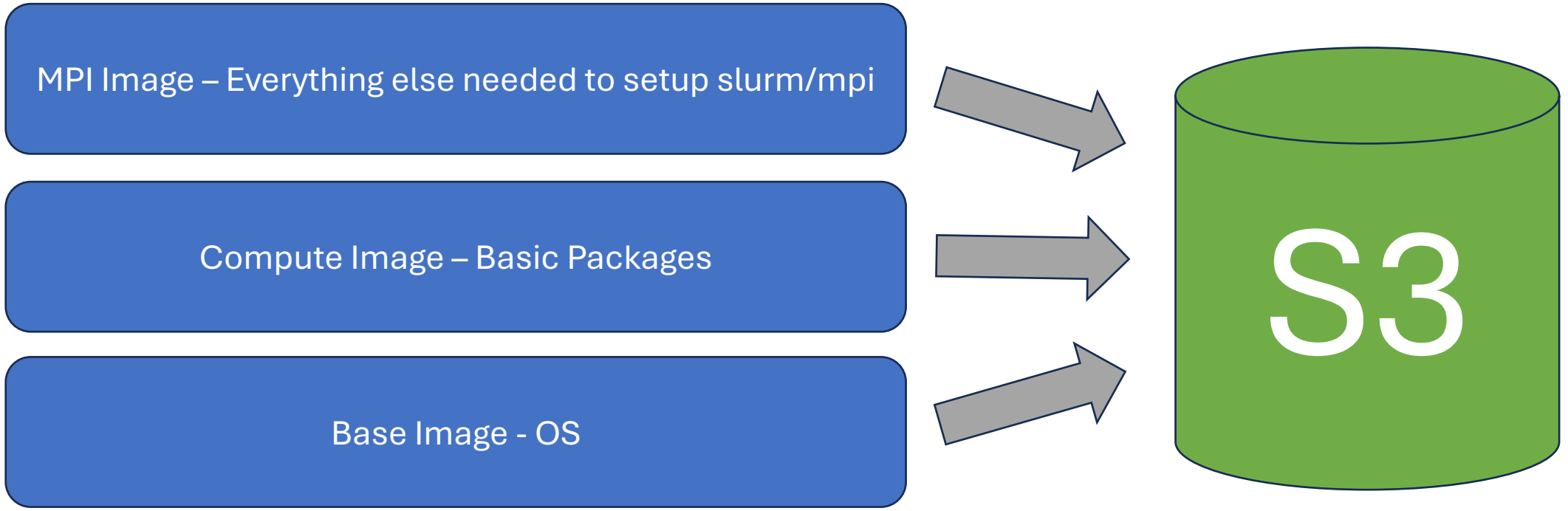
Note: OpenCHAMI doesn't include the tools to build, store, and serve system images

Prepared by LLNL under Contract DE-AC52-07NA27344.



Prepared by LLNL under Contract DE-AC52-07NA27344.

OpenCHAMI Setup- Image Building & Storage



Prepared by LLNL under Contract DE-AC52-07NA27344.

OpenCHAMI Commands We Used

Configuring the cluster's Virtual Network:

```
ig cluster set --system --default demo cluster.uri https://demo.openchami.cluster:8443
```

Populating SMD (State Management Daemon) with node data:

```
scover static -f yaml -d @/opt/workdir/nodes/nodes.yaml
```

Setting the BSS (Boot Script Service) for PXE:

```
ochami bss boot params set -f yaml -d @/opt/workdir/boot/boot-compute-debug.yaml
```

Prepared by LLNL under Contract DE-AC52-07NA27344.

Booting from OpenCHAMI is as easy as:

```
[testuser@node1 ~]$ sudo virt-install \
  --name compute1 \
  --memory 4096 \
  --vcpus 1 \
  --disk none \
  --pxe \
  --os-variant centos-stream9 \
  --network network=openchami-net,model=virtio,mac=52:54:00:be:ef:01 \
  --graphics none \
  --console pty,target_type=serial \
  --boot network,hd \
  --boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd,loader.readonly=yes,loader.type=pflash,nvram.template=/usr/share/OVMF/OVMF_VARS.fd,loader_secure=no \
  --virt-type kvm
```

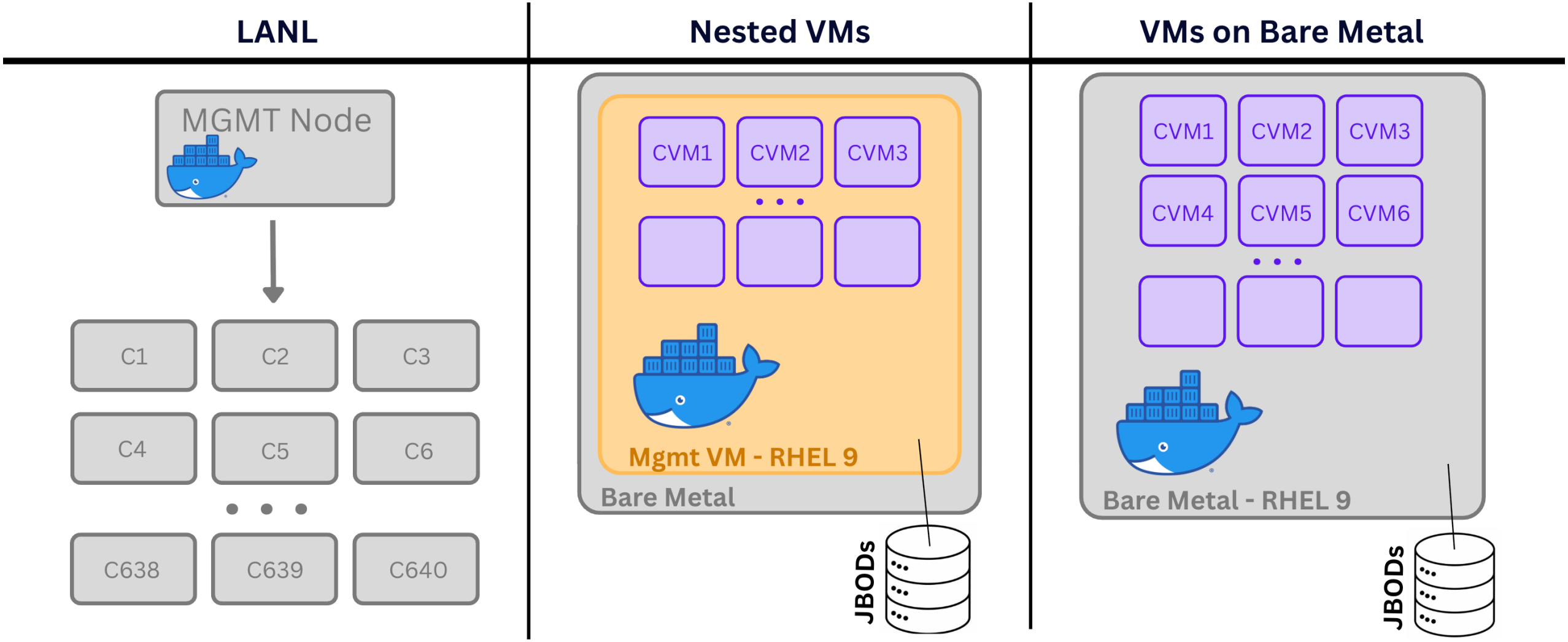
Prepared by LLNL under Contract DE-AC52-07NA27344.

So why use OpenCHAMI?

- **Cloud-like infrastructure**
  - HPC systems are rigid and require the customer to work around existing infrastructure
  - OpenCHAMI allows you to **use the resources of HPC** while allowing the **environment portability of cloud.**
  - Can be run on-prem; more control over secrets, systems kept **first-party**
- **Scientific application**
  - HPC changes constantly but OpenCHAMI can **provision older or specific environments** to recreate experiments

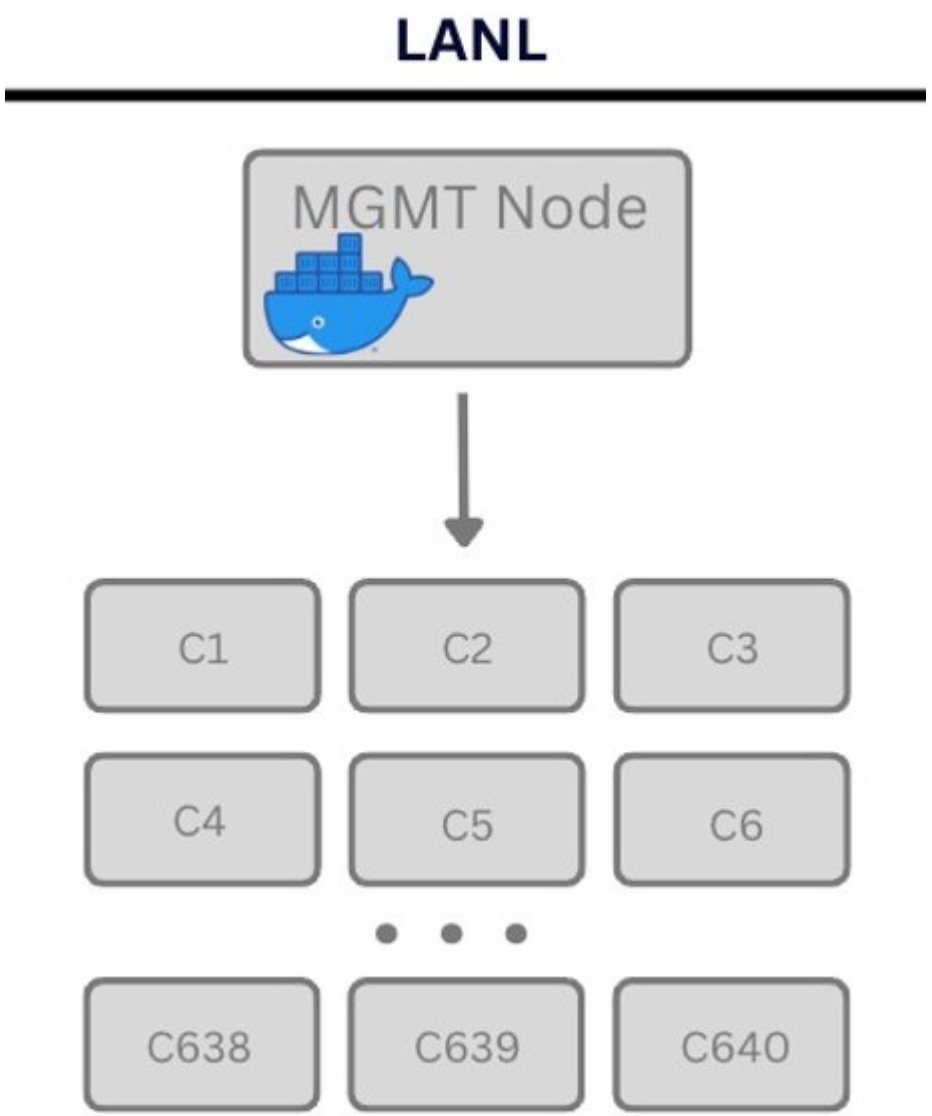
Prepared by LLNL under Contract DE-AC52-07NA27344.

Models Explored



LANL Model Analysis

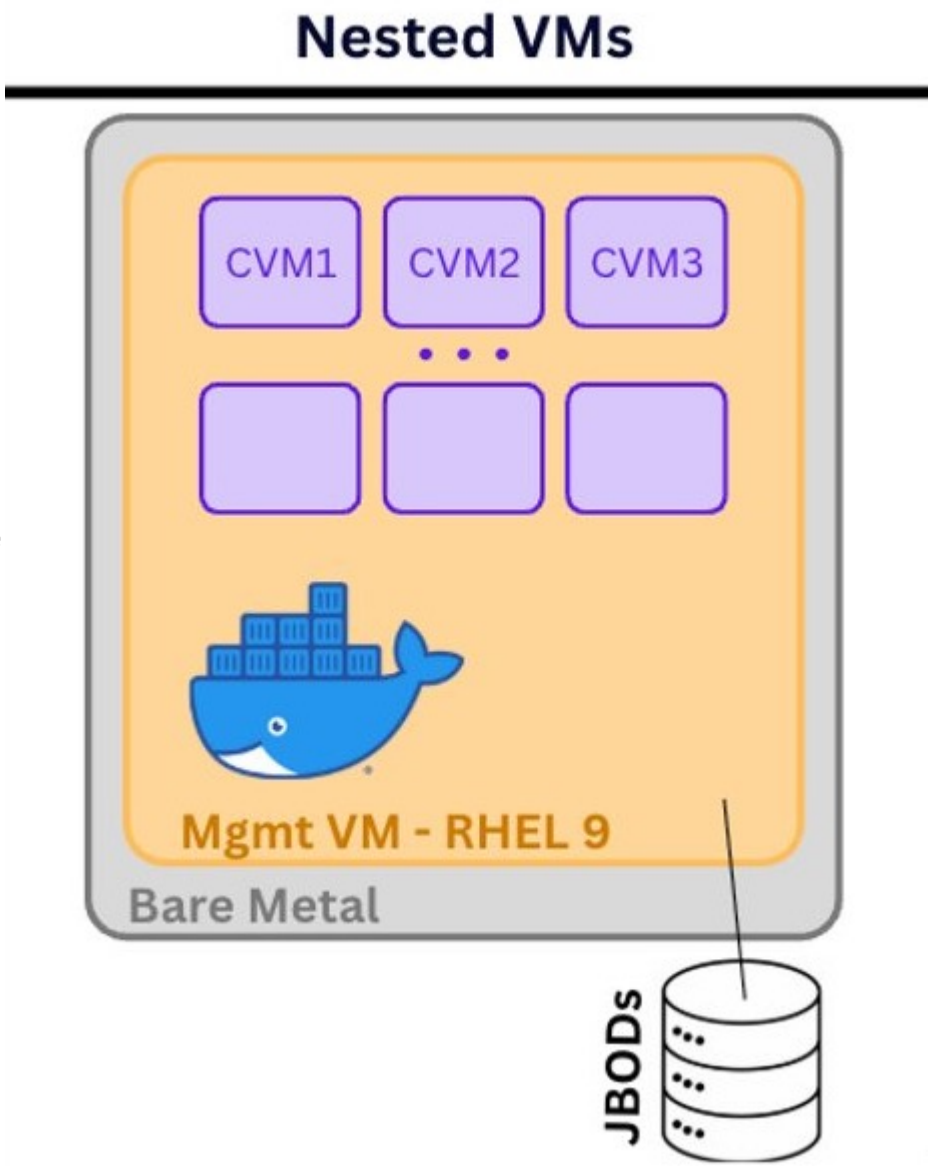
- Similar to **current HPC model**
  - Single management node provisions a large amount of **physical** compute nodes



Prepared by LLNL under Contract DE-AC52-07NA27344.

Nested VMs/Cloud Model Summary

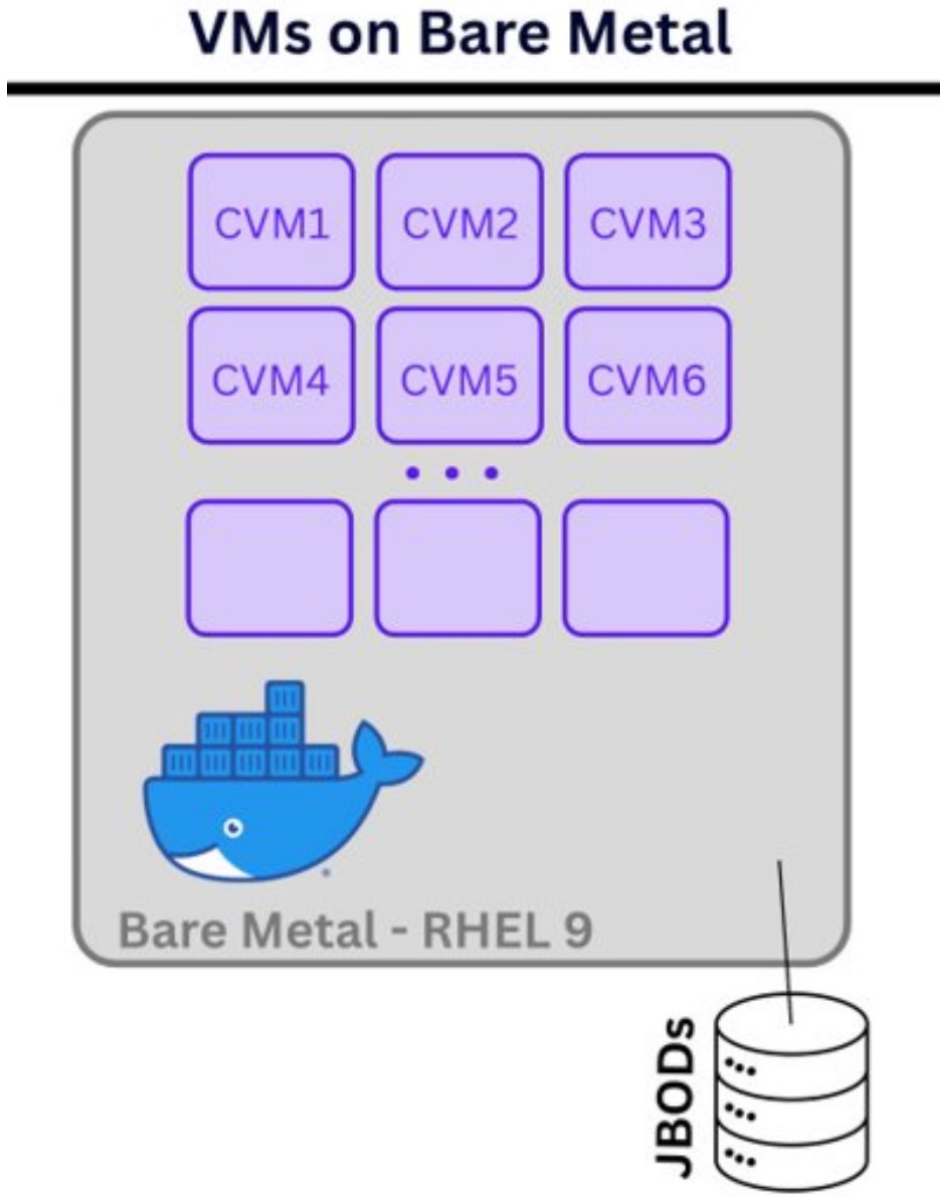
- **Advantages**:
  - Allows for **cloud-like flexibility**
  - Resource provisioning **like cloud**
  - VM parameters can be **controlled to simulate specific environments**
- **Disadvantages**:
  - Worse runtime performance
  - Larger overhead
  - Requires more **complex networking setup**
  - Can't scale out



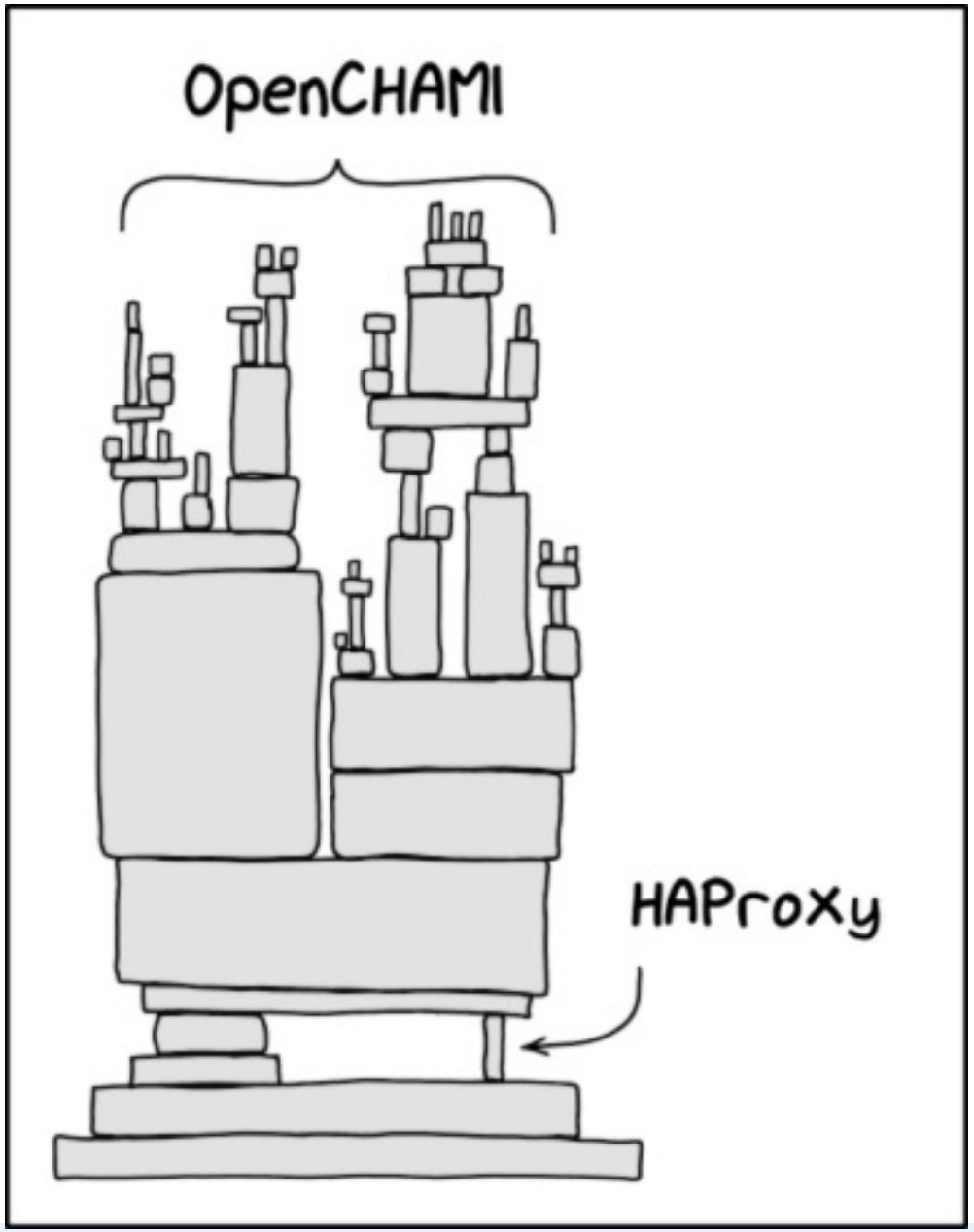
Prepared by LLNL under Contract DE-AC52-07NA27344.

VMs on Bare Metal Model Summary

- **Advantages**:
  - Simpler to provision and network
  - No nested virtualization needed (performance boost)
  - Can scale out
- **Disadvantages**:
  - Less flexible than VM environment
  - Harder to clean and reset



Prepared by LLNL under Contract DE-AC52-07NA27344.



(From XKCD)

Prepared by LLNL under Contract DE-AC52-07NA27344.

Challenges

Setting up OpenCHAMI	Using OpenCHAMI
RHEL 8 on host not supported	JWTs (JSON Web Tokens) have to be **reset every hour**
Image building temporarily **requires a lot of disk space** • The actual images aren't as big	Cloud-init makes SSH **less flexible**
Lot of dependencies that sometimes fail	Booted VMs are **diskless** and have **low storage** (~700MB)
Continuous updates to OpenCHAMI	

Next Steps

Sysadmins	Customers
Test and directly implement **LANL's** OpenCHAMI setup	**Scale out** to support more remote connections
Install and set up **FLUX**	Integrate **S3 buckets** for the provisioned nodes
Set up **High Speed Interface** (HSI)	
Setup on **TOSS 5**	

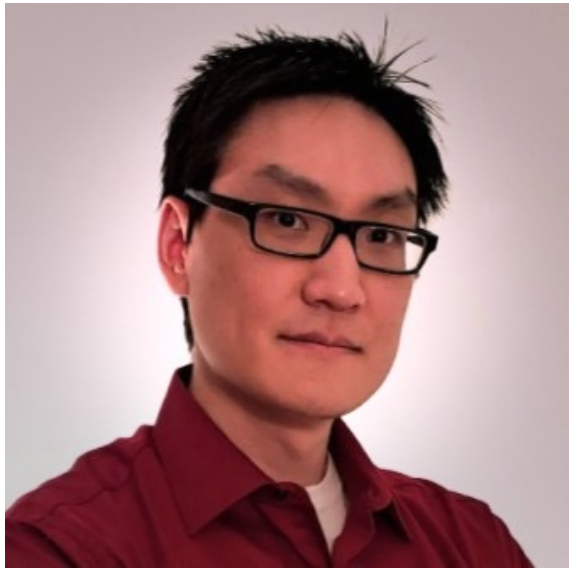
Prepared by LLNL under Contract DE-AC52-07NA27344.

Prepared by LLNL under Contract DE-AC52-07NA27344.

```
[testuser@nickel6 ~]$ sudo virt-install \
  --name compute1 \
  --memory 4096 \
  --vcpus 1 \
  --disk none \
  --pxe \
  --os-variant centos-stream9 \
  --network network=openchami-net,model=virtio,mac=52:54:00:be:ef:01 \
  --graphics none \
  --console pty,target_type=serial \
  --boot network,hd \
  --boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd,loader.readonly=yes,loader.type=pflash,nvram.template=/usr/share/OVMF/OVMF_VARS.fd,loader_secure=no \
  --virt-type kvm
```

Prepared by LLNL under Contract DE-AC52-07NA27344.

Thanks to our mentors!



Jason Kim

Naomi Cheeves

Martin Baltezore

Also, thanks to Dave Fox and Gabe Maxfield

Prepared by LLNL under Contract DE-AC52-07NA27344.

Sources

- https://www.openchami.org/
- https://github.com/OpenCHAMI/tutorial-2025/tree/main

Prepared by LLNL under Contract DE-AC52-07NA27344.