Developing with Redis

John Shiner
john.shiner@gmail.com

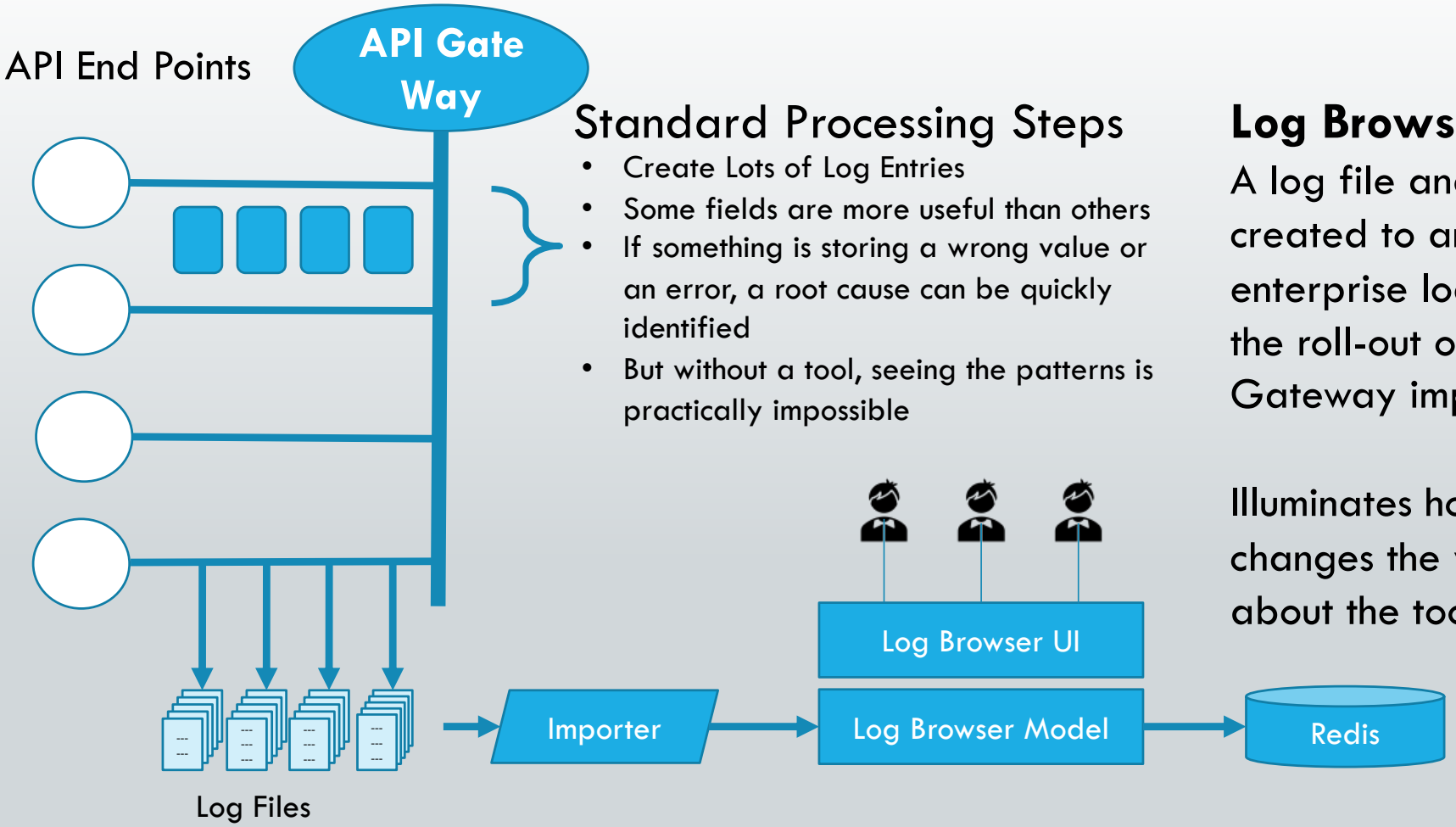
Topics

- Introduction
- The Demo Application
- Log Browser Data Model
- Primary Log Browser Views
- Application Architecture
- Development Architecture
- What Questions Are Answered?
- Demo
- Lessons Learned
- Questions and Discussion

Introduction

- Me - Better Software Better - tools, patterns, concepts, paradigm shifts, metaphors, and philosophy
  - One Example – championed DevOps practices as a means towards Complex Adaptive Development
    - Just enough structure – too rigid, the system is not adaptive
    - Too unstructured, the system is not sustainable
- Today – Demo and discuss an application that has been reimplemented to use Redis for persistence

The Demo Application – a Log Browser



Standard Processing Steps

- Create Lots of Log Entries
- Some fields are more useful than others
- If something is storing a wrong value or an error, a root cause can be quickly identified
- But without a tool, seeing the patterns is practically impossible

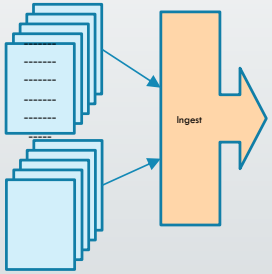
Log Browser

A log file analyzer tool created to analyze enterprise log files during the roll-out of an API Gateway implementation

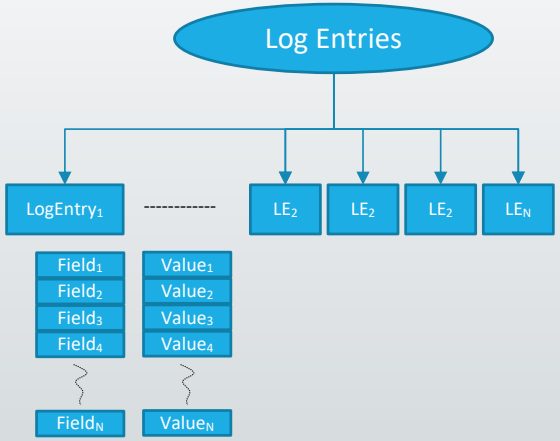
Illuminates how Redis changes the way to think about the tool architecture

Log Browser Data Model

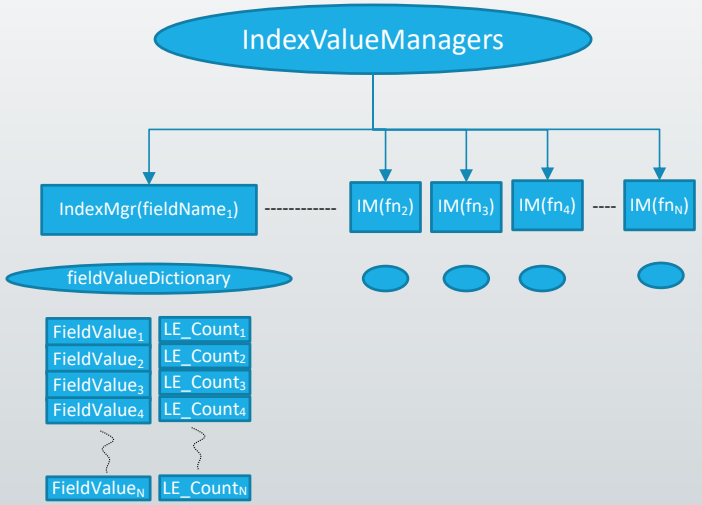
Log Files



Digitized Log Entries



Selected Index Analysis



Other Redis keys

- loadedLogFiles
- contentDirtyFlag
- vmDirtyFlag

LogEntry (each row in a log file)

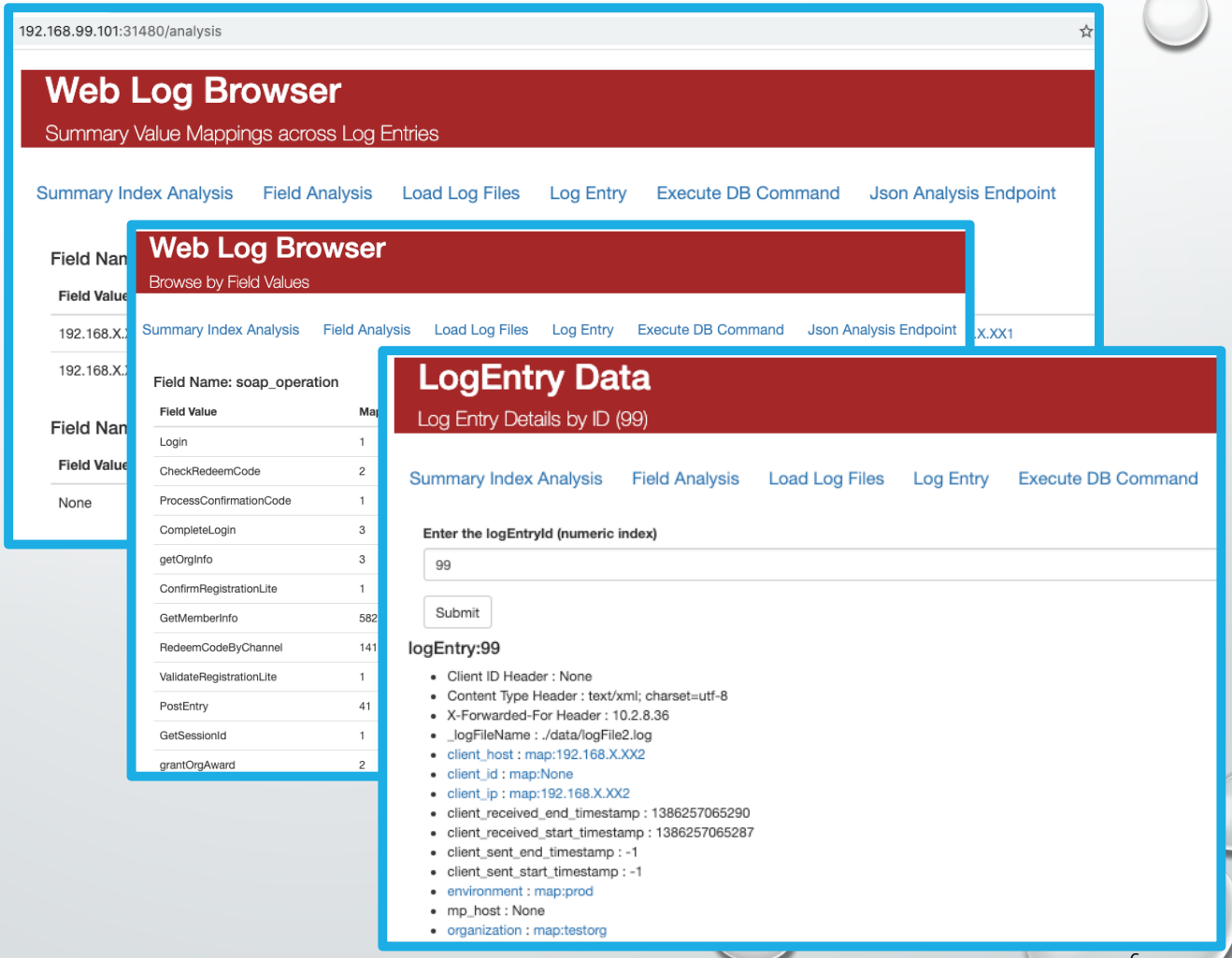
Redis Hash
key === **logEntry:<id>**

Indexed Field <u>Value</u> mapping to Log Entries

Redis Ordered Set containing each mapped Log Entry ID
key === **map:<fieldName>:<fieldValue>**

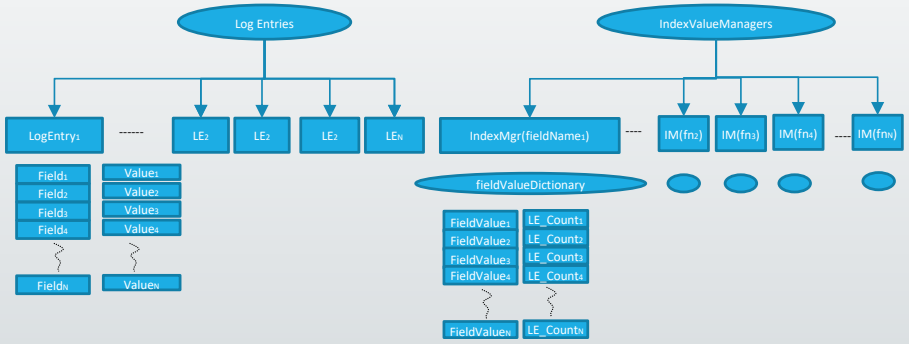
Primary Log Browser Views

- **Summary Value Mappings**
- **Summary by Index**
- **LogEntry Detail View**

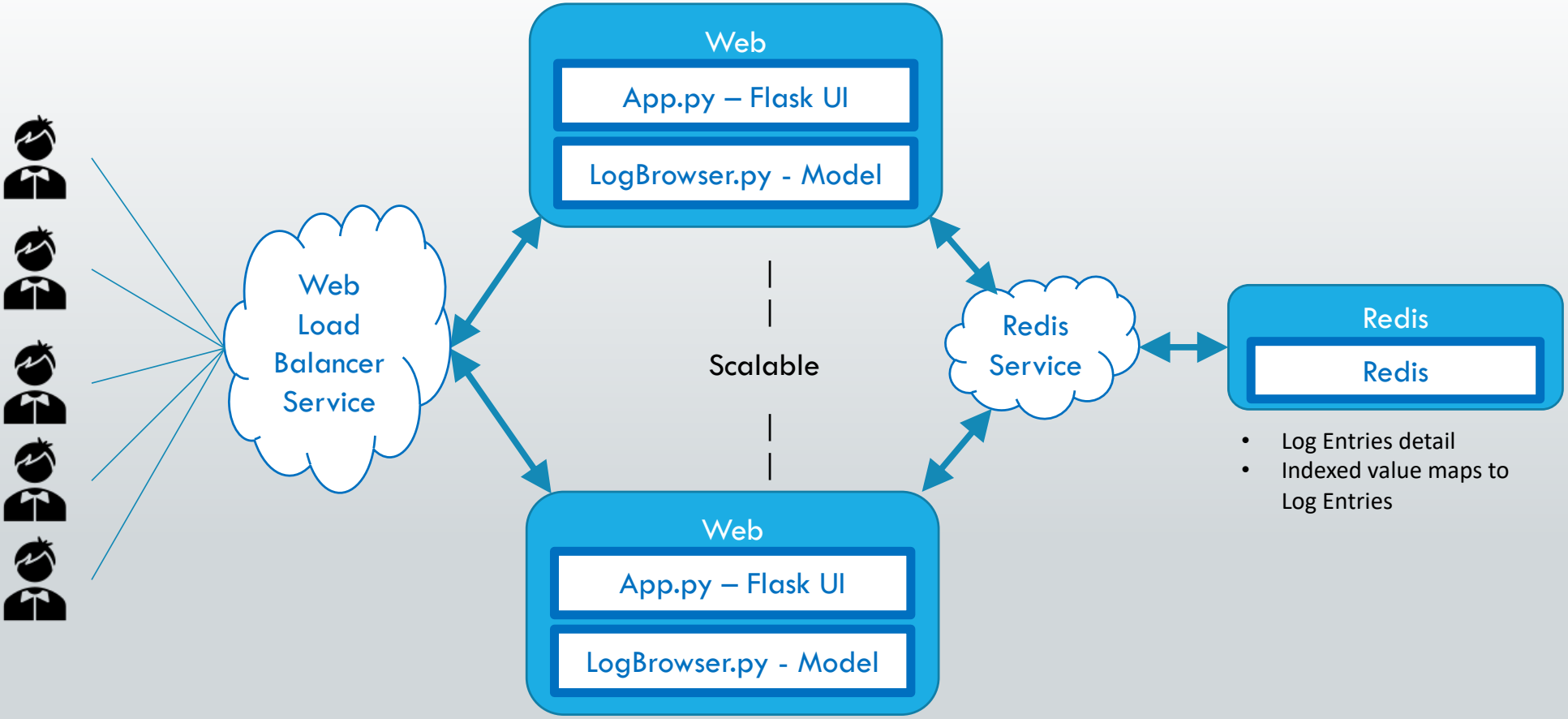


What Questions Do the Data Model Answer?

- What are the range of values for each field?
- What is the log entry distribution (count and percentage) across the range of values for each indexed field?
- How do you view the entire record for a log entry that has set a particular field value?
- What log file contained a specific log entry?
- What other log entries match a field value that this log entry has set?
- What are the other field values that have been set for this field?

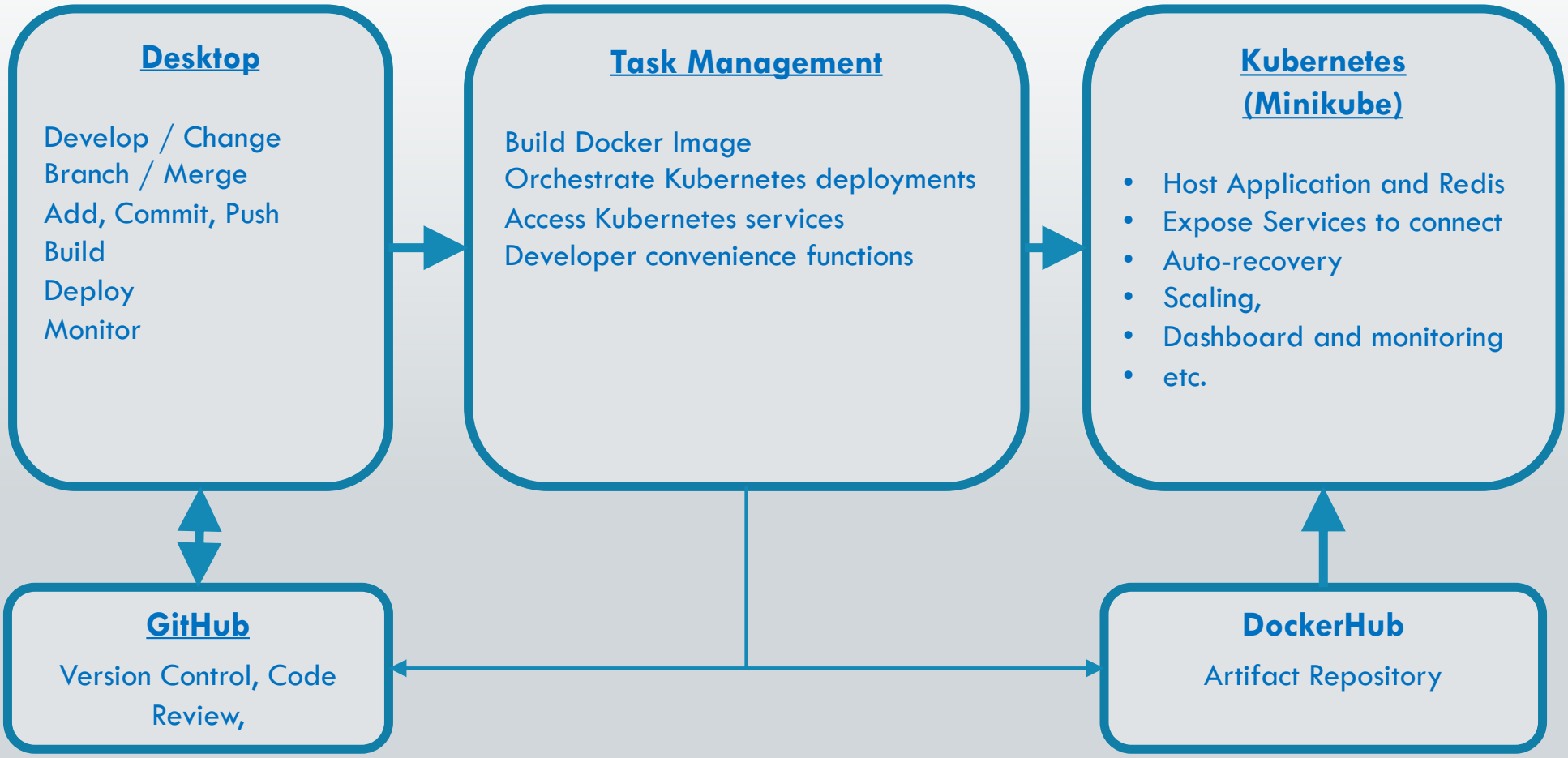


Application Architecture



- Log Entries detail
- Indexed value maps to Log Entries

Development Architecture



Demo

- Orient with Desktop, file structure, project code
- Task automation - Invoke commands, tasks.py
- Kubernetes - k get pods
- inv clean-deploy (undeploy, rmi, build docker image, deploy, and launch)
- k get pods
- inv scale –n 5
- inv dash
- inv gh
- db/info, json

- Redis commands (client)
  - info
  - keys map:soap_operation:*
  - hgetall logEntry:1566
  - zrange map:soap_operation:UpdateAward 0 -1
  - smembers loadedLogFiles
- Code change, rebuild/deploy
  - change indexNames comments in LogBrowser for different indexed values

Key Takeaways

A conceptual shift

- Redis has a very low-barrier to entry.
  - More use of conventions and patterns to address types of storage and durability tradeoffs
    - Index naming, wildcard scan selects
  - Module capabilities are exciting and promise to accelerate the next generation of features
- Break habit of creating data structures to manage persistence
  - Just have your object accessors store the data in Redis as data structures
- Redis is *modern application* friendly
  - Plays well with Docker, Kubernetes, REST
  - Hence, easily scales, recovers, provides status, etc. under infrastructure-as-code conventions for robust, repeatable, portable installations

Appendix

Log Browser Data Model Described

Log Files

Ingest

Analysis can span any number of log files

Digitized Log Entries

LogEntries

LogEntry

Redis Hash
logEntry:<id>

Fields
- fieldName
- fieldValue

fields: x1, x2, ..., xN
values: y1, y2, ..., yN

Each Row in a Log File becomes a **Log Entry**
A log entry has *standard field names*
A log entry may have *various field values*

Plus other Redis keys
- loadedLogFiles
- contentDirtyFlag
- vmDirtyFlag

Selected Index Analysis

IndexValueManagers

Selected Indices are a subset of the available fields deemed important for analyzing API deployment errors

IndexMgr(fieldName)
- fieldName
- table_content
- value_map

Selected Indices become a data structure (an IndexMgr) to map field names to the range of values for that field across all of the log entries processed

fieldValueDictionary
- fieldValue
- mappedLogEntries

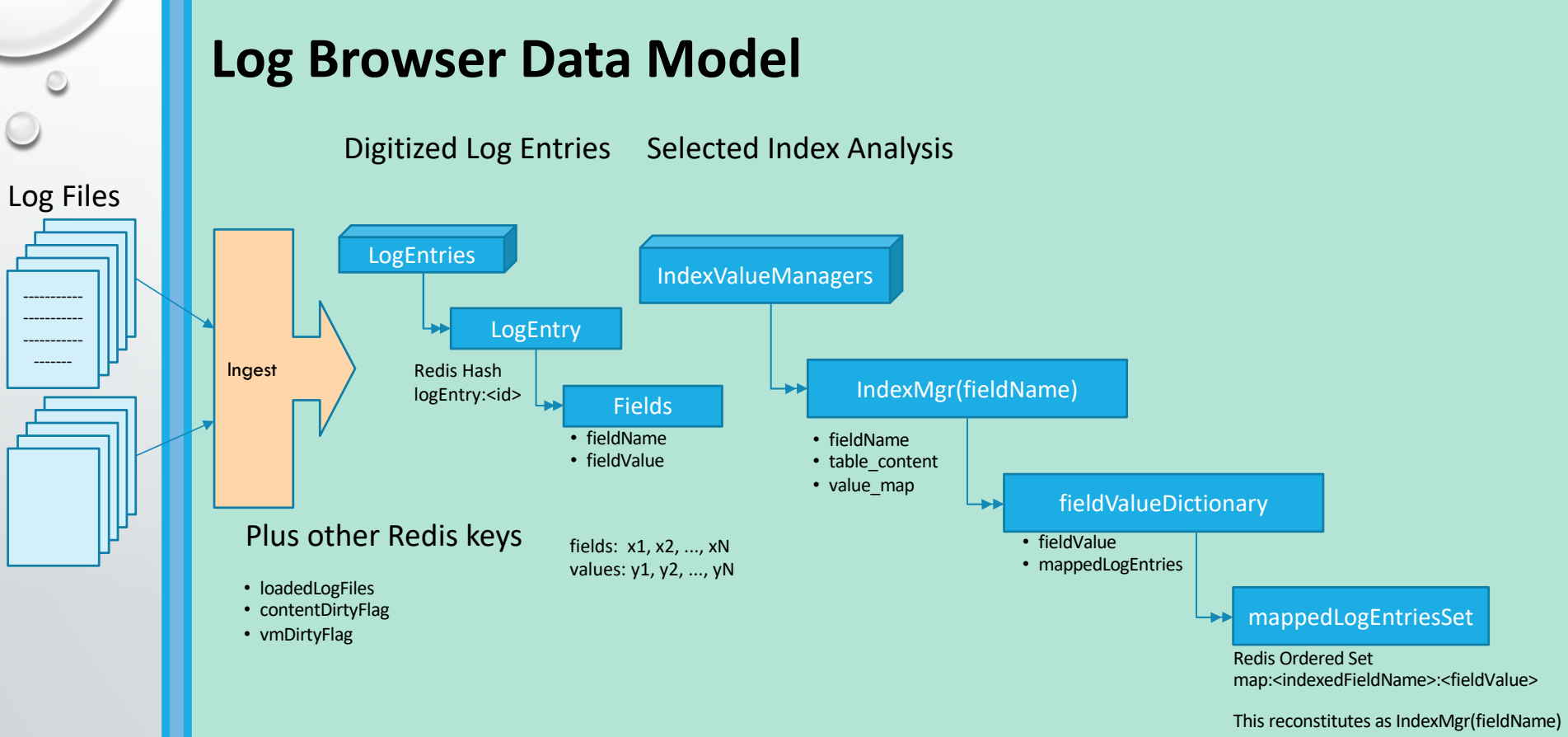
An IndexValueManager is accessed by it's selected index field name and holds a value map, for each of the values assigned to that field, to a key in Redis containing an ordered set of log entries that set that value for that field.

mappedLogEntriesSet

Redis Ordered Set
map:<indexedFieldName>:<fieldValue>

This reconstitutes as IndexMgr(fieldName)

Log Browser Data Model



Demo Objectives

- Reimplement a log file analyzer using Redis
- The *LogBrowser* Application –hands-on practice using
  - Python3, GitHub, Flask, Redis, a Python task manager (invoke), and Kubernetes
  - Harvest lessons learned
  - Create a platform with a roadmap to provide more hands-on experiences
- Leverage development best practices while learning how Redis fits
  - Task automation; git branching and merging; docker containerization; kubernetes deployment, monitoring, and resiliency features (via Minikube)
- Harvest lessons learned and create a capability backlog

Project Repository Structure

Log File Staging
data/

Flask User Interface
app.py
templates/

Application Model
LogBrowser.py

Configuration, Packaging, Deployment
config.py
Dockerfile
docker-compose.yml
k8s/

Development Automation
requirements.txt
tasks.py

Task Automation Example: invoke undeploy rmi build deploy

Command	Description
build	Build the web docker image
clean-deploy	Undeploy, remove the web docker image from local registry, build a web docker image, and deploy
dash	Run this to launch the minikube dashboard
db	Run the output of this command for a parameterized Redis-cli command string
dbport	Run this to return the exposed port for the redis service
deploy	Run this to deploy the application stack to minikube
gh	Open the current github branch on GitHub
push	Run the bin/push-to-dockerhub script -- requires env vars
rmi	Remove the web image from local (minikube) docker registry
scale	Run this to scale the web pods to <num> replicas
st	Open the current repository in Sublime Text
undeploy	Run this to remove (all) the application stack(s) from minikube
usage	Example: inv undeploy rmi build deploy
web	Launch the minikube hosted 'web' service to run the deployed application
webport	Run this to return the exposed port for the web service