basefarm

# PART 1: NEXT-GEN EHEALTH PLATFORM
# PART 2: WHY CONTAINERS & MICROSERVICES

Bent Terp
Senior Solutions Architect
Basefarm Group

basefarm

# PART 1: CUSTOMER CASE
# BUILDING A NEXT-GEN PLATFORM FOR EHEALTH SERVICES

Bent Terp
Senior Solutions Architect
Basefarm Group

# CUSTOMER CASE

**“A large Swedish ehealth provider”**

- Dozens of ehealth services
- On a national level
- To both healthcare organizations and patients

# MULTIPLE SERVICES

- Separate administrative teams
- Independent consultants as developers
- Bring-Your-Own architecture
- Developers without access to production

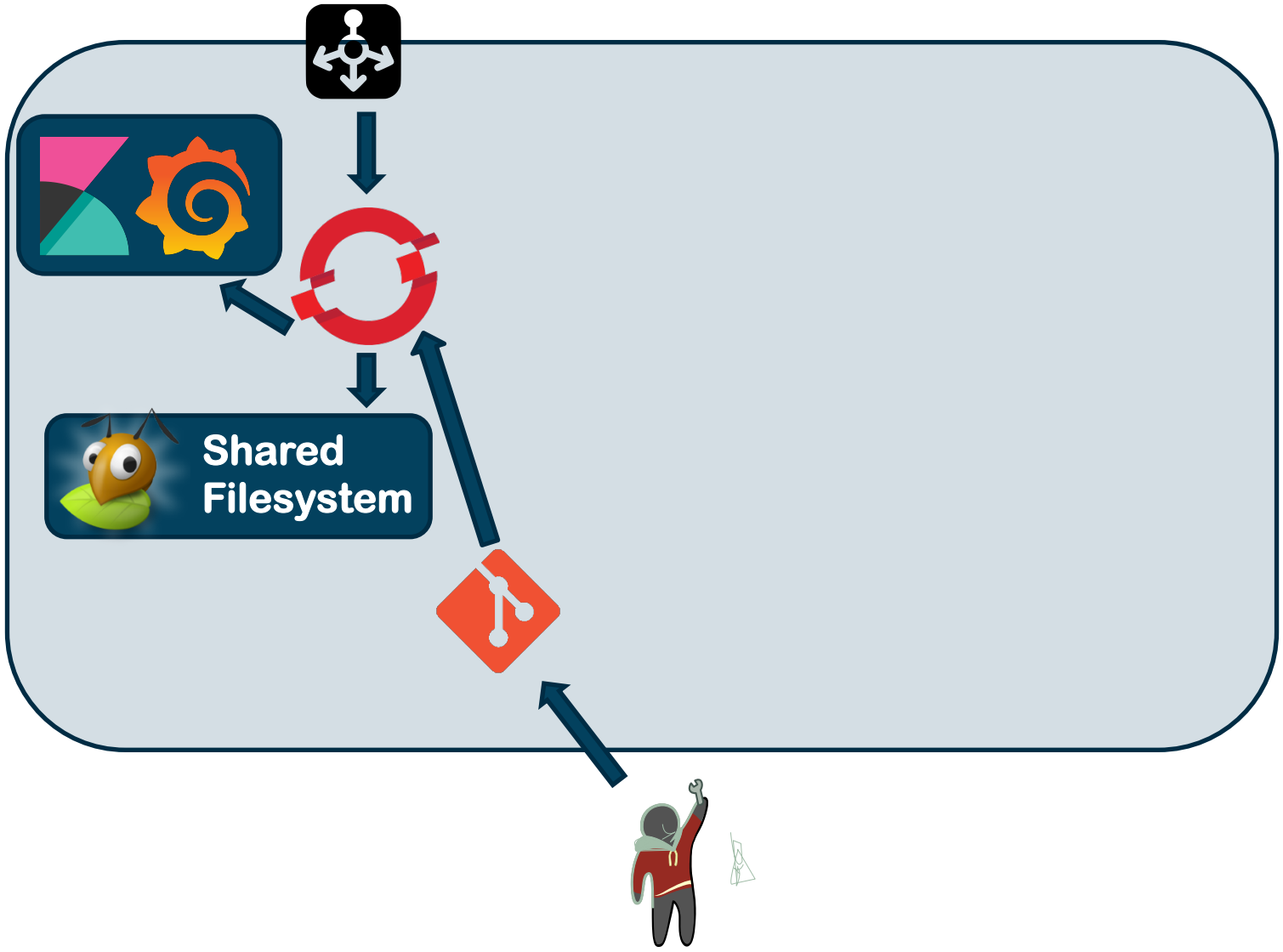
# SINGLE OPERATIONS TEAM

- Diversity in procedures, ie deploy
- Lack of control-plane visibility
- SLA: 99.995%
- No planned downtime

# ADDITIONAL GOALS

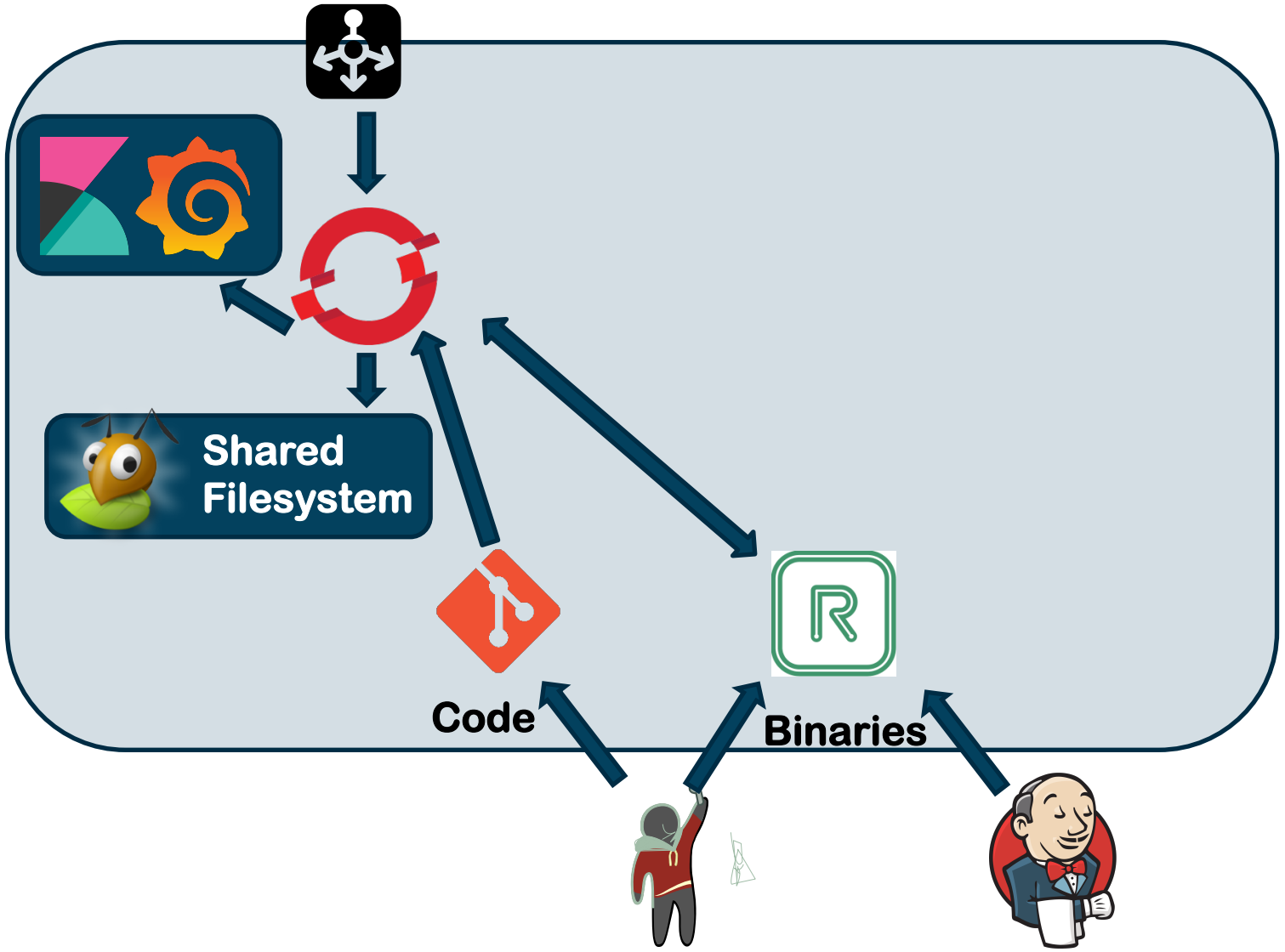
- Take control of the source code
- Protect investment in existing pipelines
- Unified admin console of all services
- Facilitate recruitment

# STARTING OUT



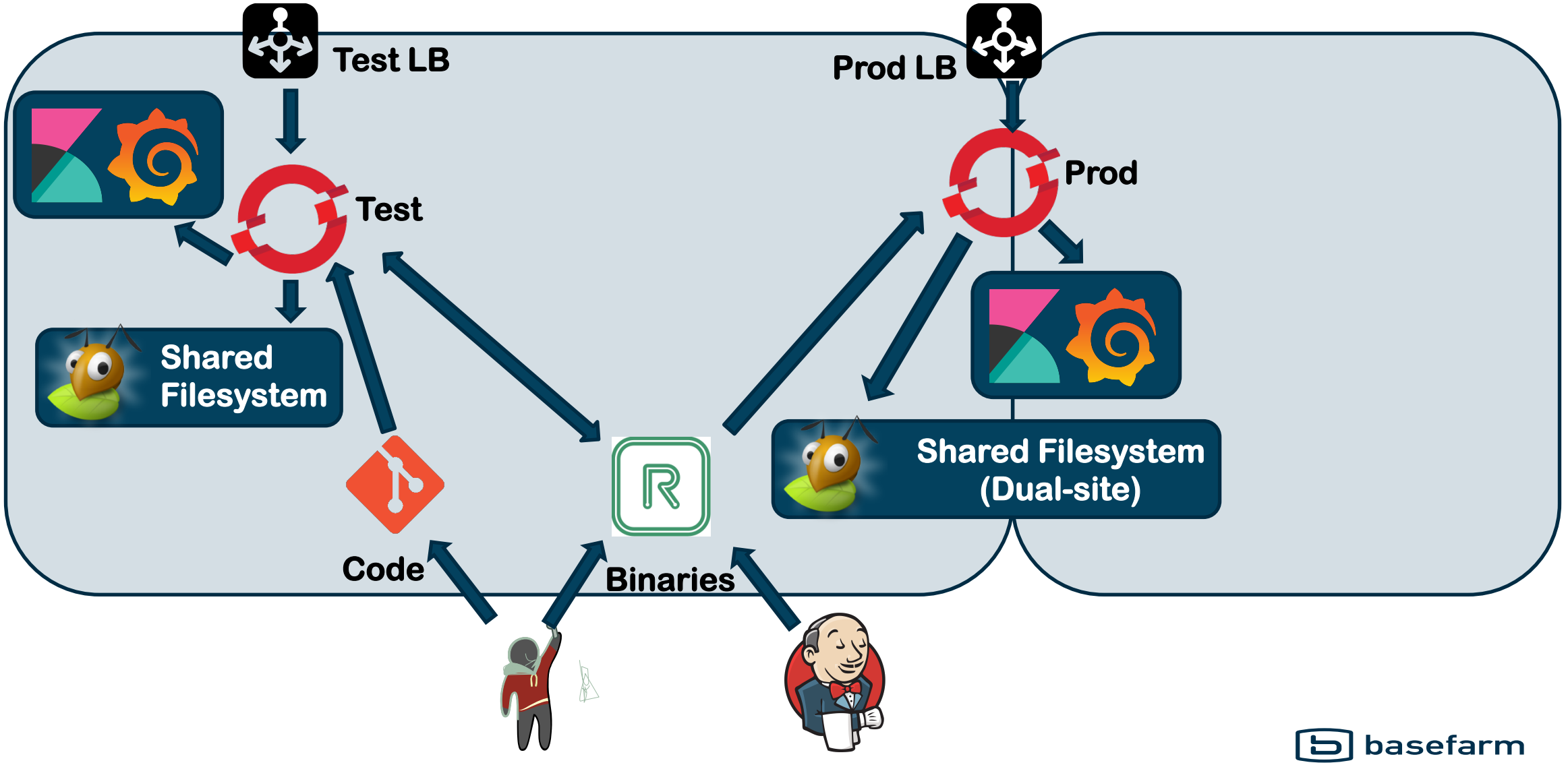

basefarm

# WE HAVE SOME BINARIES ALREADY BUILT



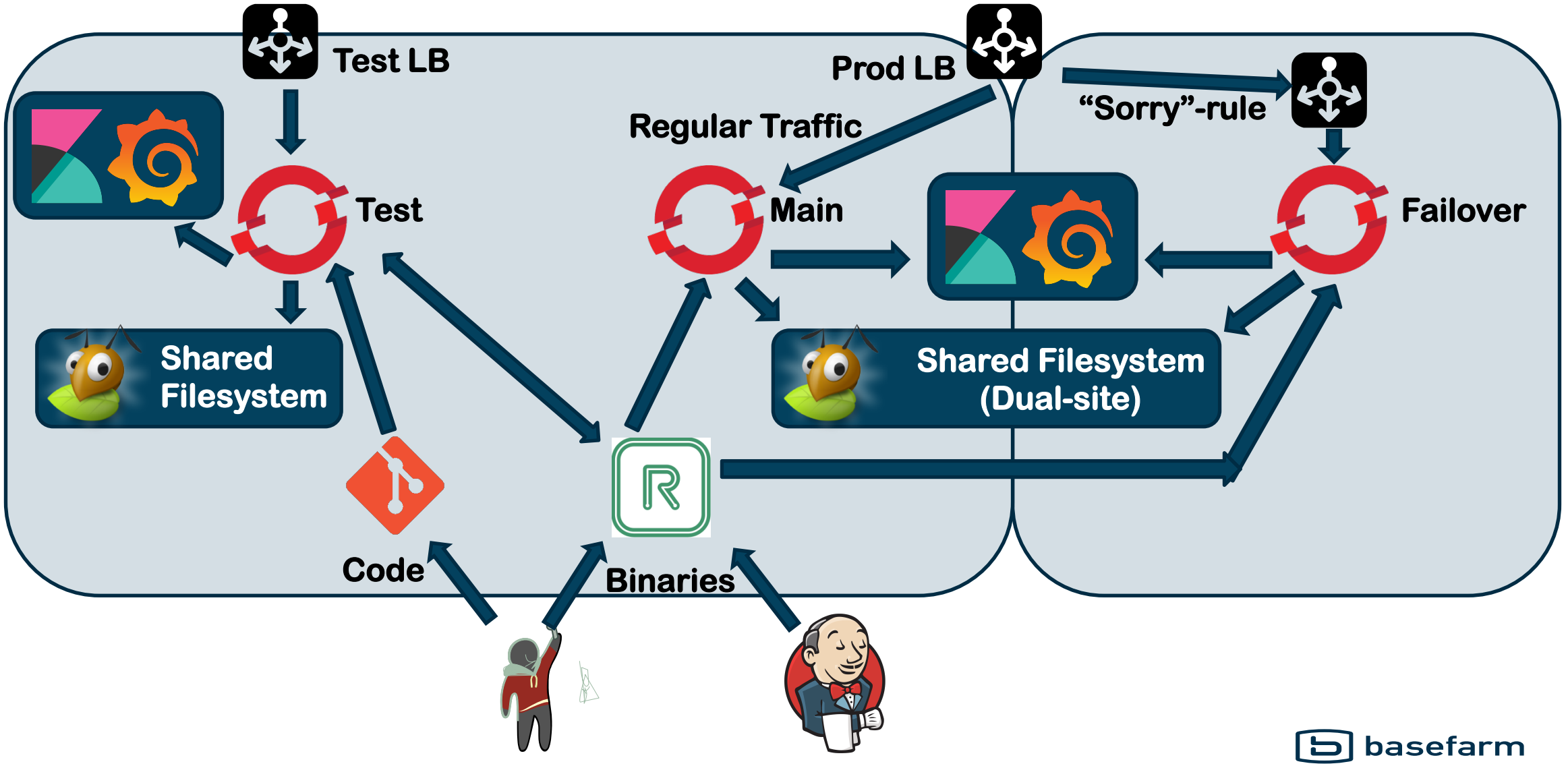

basefarm

# PROD SEPARATE FROM TEST – AND DUAL-SITE



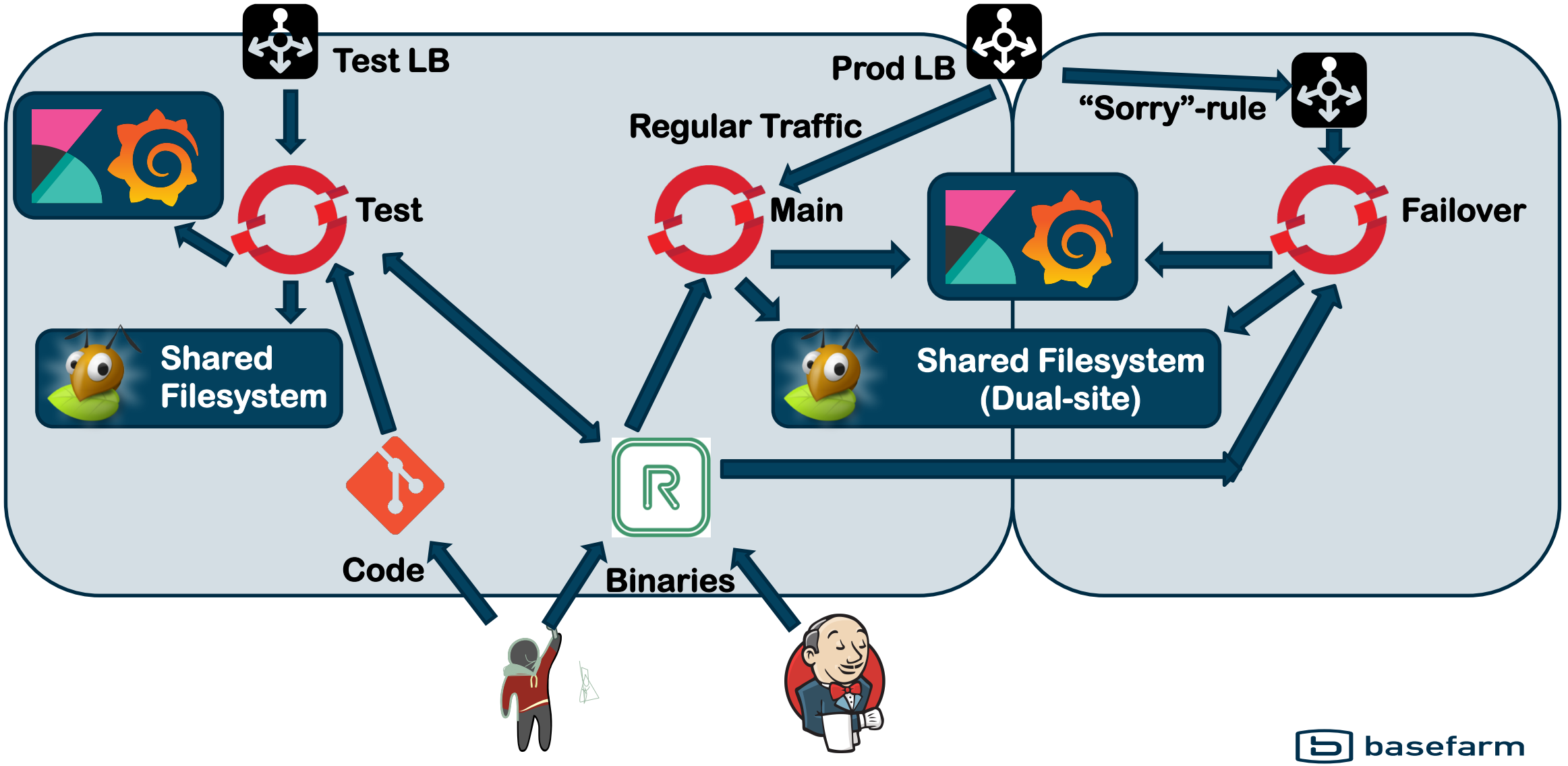

basefarm

# CLUSTER MAY GO DOWN, BUT NOT SERVICE




basefarm

# END RESULT…. SO FAR




basefarm

basefarm

# PART 2: WHY DO AGILE & DEVOPS LEAD TO MICROSERVICES & CONTAINERS

This is my story, told from my point of view.
Your story will be different – that's okay

# CONFLICTING REQUIREMENTS

DevOps teams should be able to “do everything themselves”

Agile teams should be small for efficiency

We need to reduce “everything”

# REDUCING “EVERYTHING”

- Narrow scope
  - Divide into microservices
- Simplify operations
  - Deploy using container platform

basefarm

# MICROSERVICE ARCHITECTURE

- Decoupling complexity
- Separation at network layer
- Independence
  - release cycles
  - programming language
- Separation of API and UX

# REDUCING TIME WASTED

| Monolith | Sprint | Release | Sprint | Release | Sprint | Release |
|---|---|---|---|---|---|---|
| Team A | | | | | | |
| Team B | | | | | | |
| Team C | | | | | | |
| | | | | | | |
| **µService** | | | | | | |
| µSvc A | | | | | | |
| µSvc B | | | | | | |
| µSvc C | | | | | | |
| | | | | | | |

DISCLAIMER:
GROSS OVER-
SIMPLIFICATION

basefarm

# YOUR API IS A PROMISE

Keep your promises!

- Adding features
- Invisible changes
- Versioning
- Life-Cycle

# EVOLUTION IN DEPLOYMENT/PACKAGING

1. **On Premise**
   … Your servers, your basement

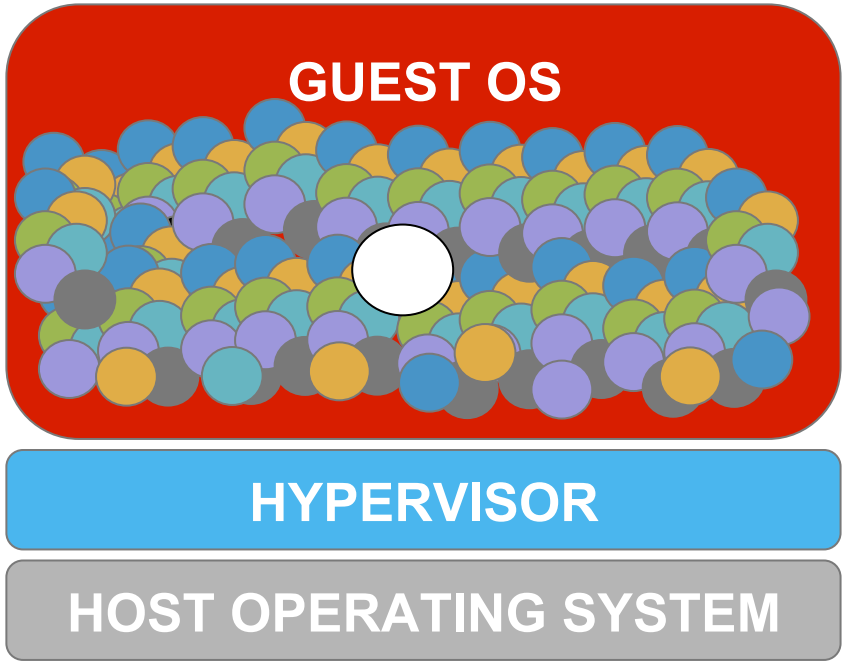
basefarm

# EVOLUTION IN DEPLOYMENT/PACKAGING

1. **On Premise**
   … Your servers, your basement
2. **Co-location**
   … Your hardware in somebody else's DC

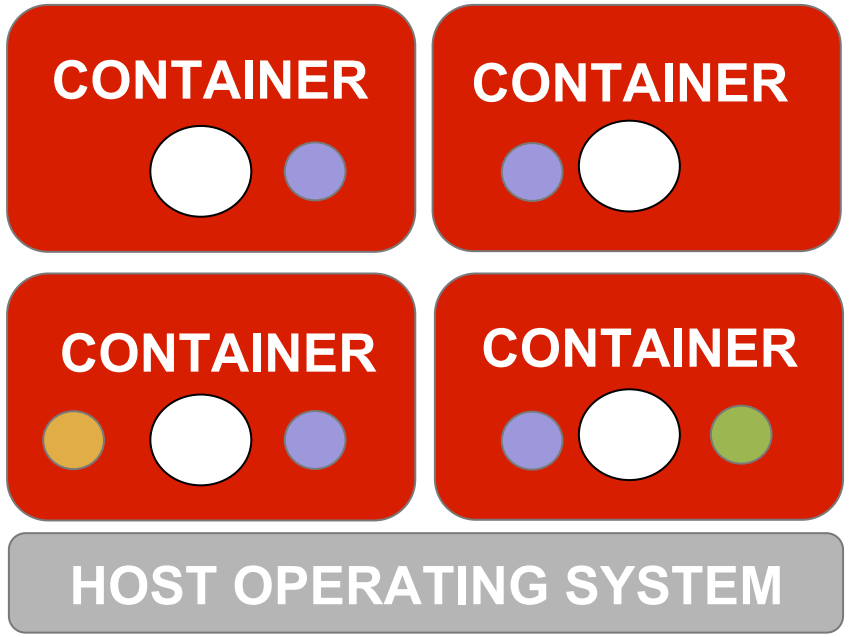
# EVOLUTION IN DEPLOYMENT/PACKAGING

1. **On Premise**
   … Your servers, your basement
2. **Co-location**
   … Your hardware in somebody else's DC
3. **Virtual Servers**
   … Your OS on somebody else's hardware



# EVOLUTION IN DEPLOYMENT/PACKAGING

1. **On Premise**
   … Your servers, your basement
2. **Co-location**
   … Your hardware in somebody else's DC
3. **Virtual Servers**
   … Your OS on somebody else's hardware
4. **Containers**
   … Your application in somebody else's OS




basefarm

# LOTS TO CONSIDER

Orchestration

Build Automation

Deployment Automation

Change management

Self Service

Logs & Metrics

Compliance

Monitoring

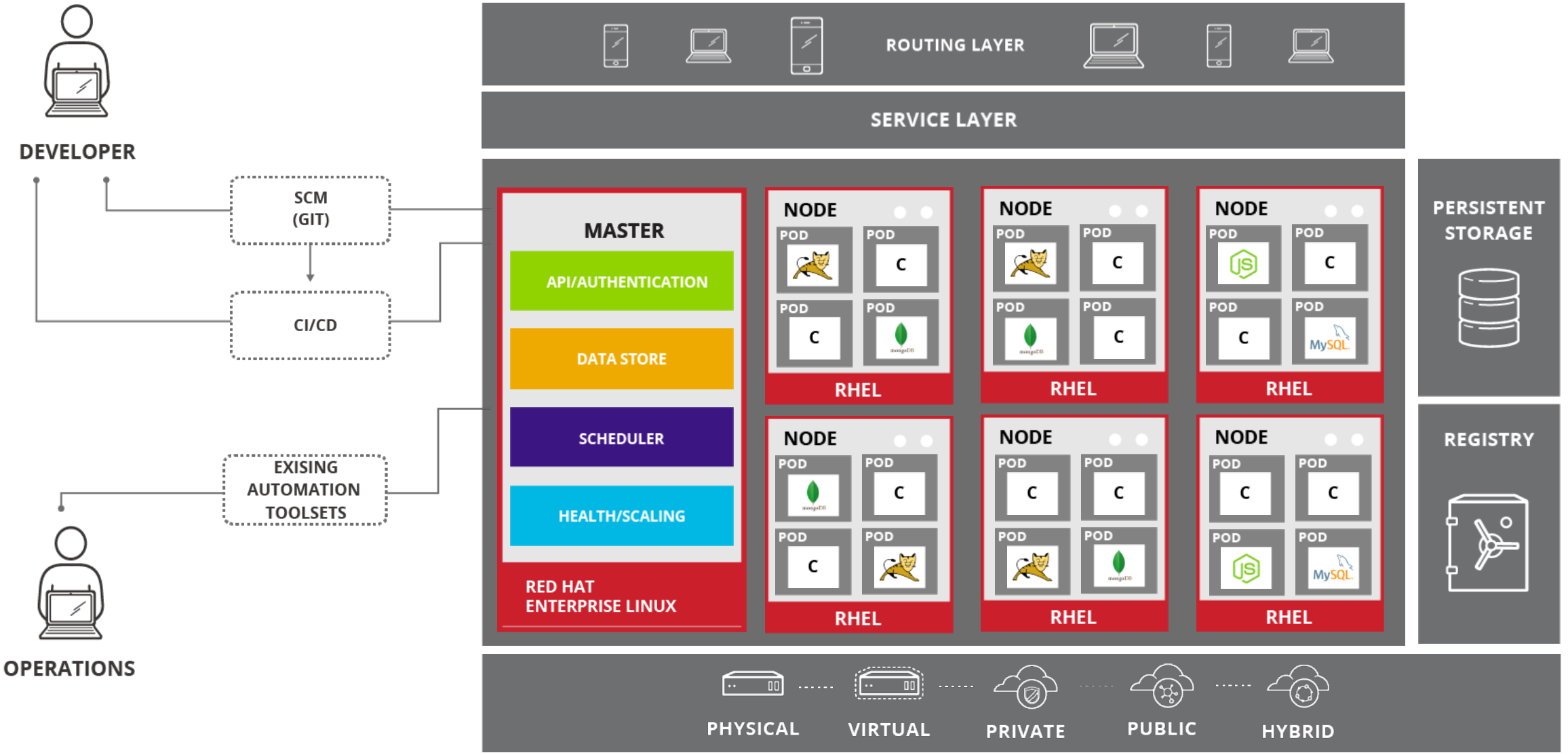
Operational Management

Security

# WE ARE HERE TO HELP YOU!

basefarm

# LESS TO CONSIDER




basefarm

# STILL LEFT TO CONSIDER

Processes

People

# TREAT CONTAINERS AS IMMUTABLE

Deployment

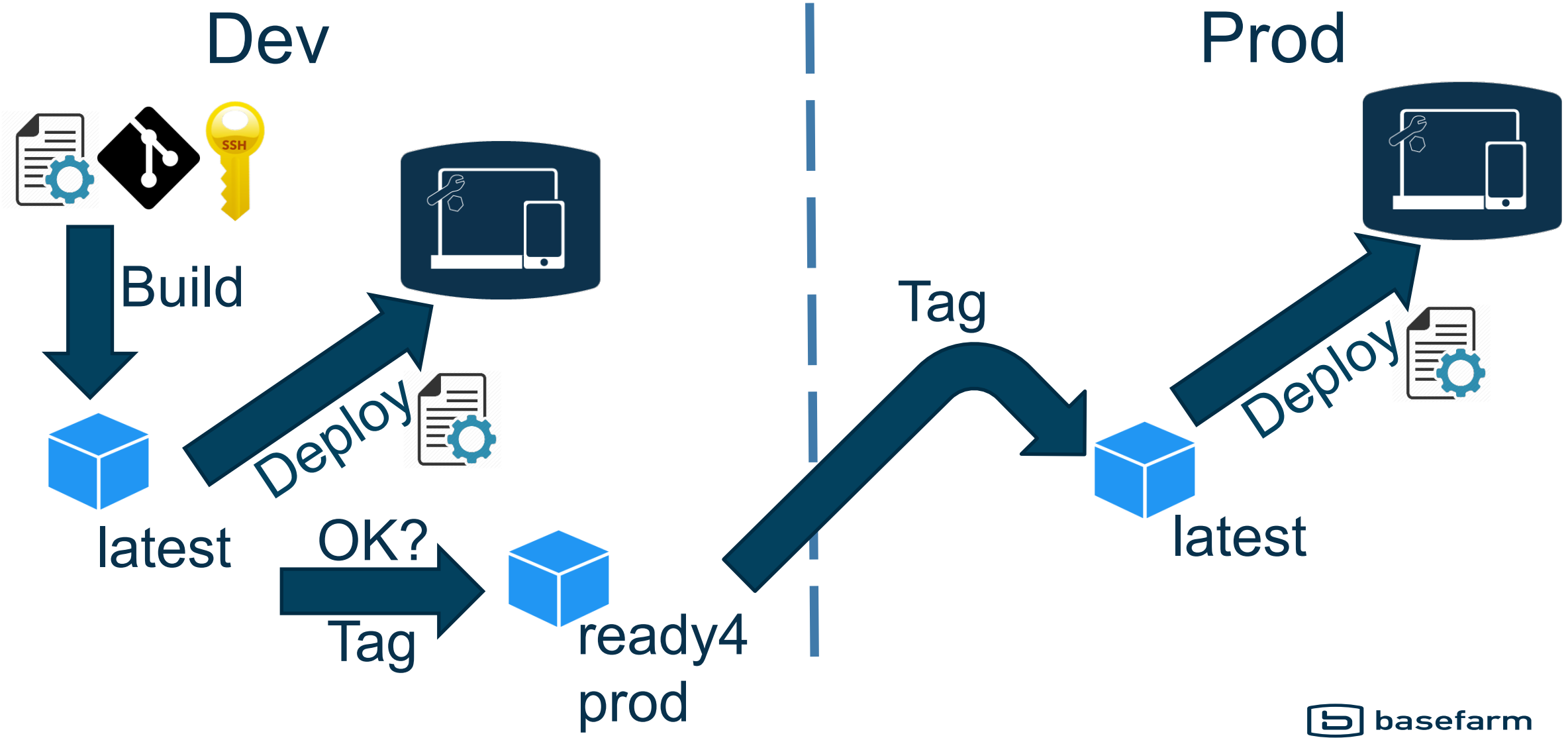
Containerized Application

Application Properties

Infrastructure Configuration

basefarm

# ”SAFELY AGILE”




basefarm

# EVOLUTION IN DEPLOYMENT/PACKAGING

3. **Virtual Servers**
   … Your OS on somebody else's hardware
4. **Containers**
   … Your application in somebody else's OS
5. **Serverless**
   … Your code in somebody else's container
6. **AI?**
   … Let the code write itself to fit the data model?