

# EIGHT STEPS TO CLOUD-NATIVE APPLICATION DEVELOPMENT

## WHAT IS CLOUD-NATIVE APPLICATION DEVELOPMENT?

Cloud-native application development is an approach to building and running applications. It speeds time to market by using the cloud computing model, which is based on these key tenets:

**SERVICE-BASED ARCHITECTURE**  
Uses modular, loosely coupled services, such as microservices. Increases development speed without increasing complexity.

**CONTAINER-BASED INFRASTRUCTURE**  
Uses containers as a common operational model across application technology stacks, offering portability, horizontal scaling, and automation with low overhead and high density.

**DEVOPS PROCESSES**  
Follows agile methodology, which builds and delivers applications collaboratively.

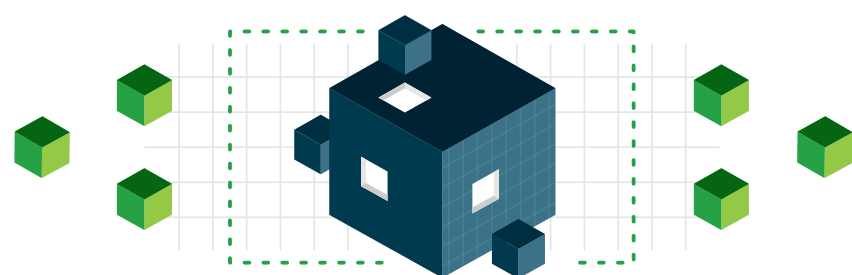
**API-DRIVEN COMMUNICATION**  
Uses lightweight application programming interfaces (APIs) that reduce the complexity and overhead of deployment, scalability, and maintenance. Composes new business capabilities and opportunities with the exposed APIs.

## 8 STEPS

## RECOMMENDATIONS TO HELP YOU SUCCEED IN CLOUD-NATIVE APPLICATION DEVELOPMENT

### STEP 01 EVOLVE A DEVOPS CULTURE AND PRACTICES

Take advantage of new technology, faster approaches, and tighter collaboration by embracing the principles and cultural values of DevOps and organizing your organization around those values.

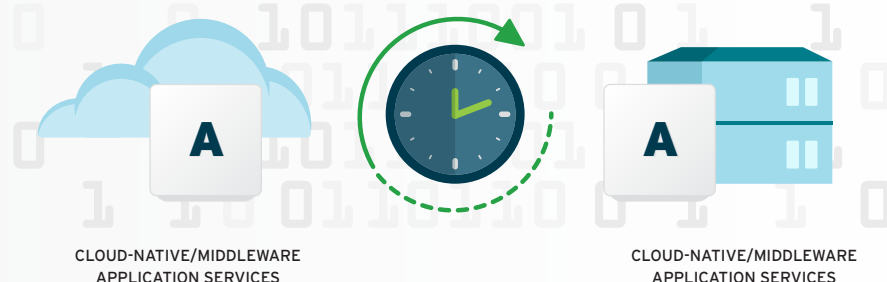


### STEP 02 SPEED UP EXISTING APPLICATIONS USING FAST MONOLITHS

Accelerate existing applications by migrating to a modern, container-based platform—and break up monolithic applications into microservices or miniservices for additional efficiency gains.

### STEP 03 USE APPLICATION SERVICES TO SPEED UP DEVELOPMENT

Speed software development with reusability. Cloud-native application services are ready-to-use developer tools. However, these reusable components must be optimized and integrated into the underlying cloud-native infrastructure to maximize benefits.



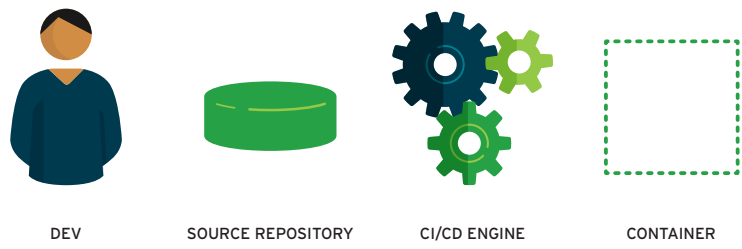
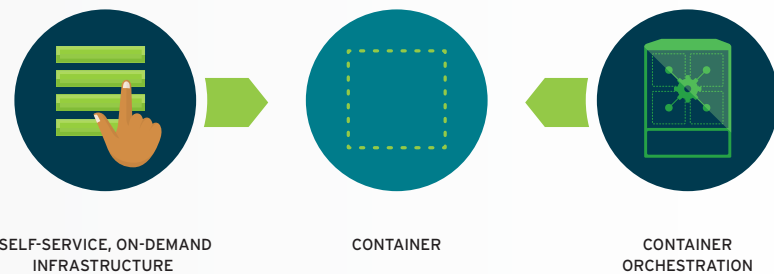
SPRING BOOT	DROPWIZARD	NODE.JS
ECLIPSE MICROPROFILE	PYTHON	GOLANG
ECLIPSE VERT.X	APACHE OPENWHISK	JAKARTA EE

### STEP 04 CHOOSE THE RIGHT TOOL FOR THE RIGHT TASK

Use a container-based application platform that supports the right mix of frameworks, languages, and architectures—and can be tailored to your specific business application need.

### STEP 05 PROVIDE DEVELOPERS WITH SELF-SERVICE, ON-DEMAND INFRASTRUCTURE

Use containers and container orchestration technology to simplify access to underlying infrastructure, give control and visibility to IT operations teams, and provide robust application life-cycle management across various infrastructure environments, such as datacenters, private clouds, and public clouds.



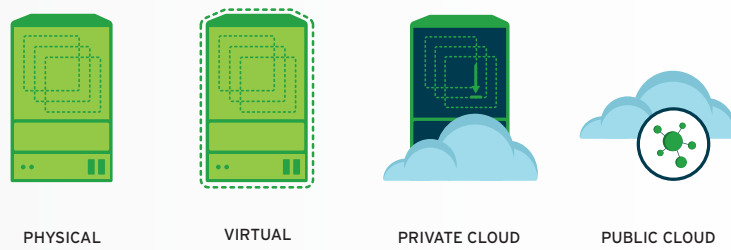
### STEP 06 AUTOMATE I.T. TO ACCELERATE APPLICATION DELIVERY

Lay the foundation for IT automation with:

- Automation sandboxes for learning the automation language and process.
- Collaborative dialog across organizations for defining service requirements.
- Self-service catalogs that empower users and speed delivery.
- Metering, monitoring, and chargeback policies and processes.

### STEP 07 IMPLEMENT CONTINUOUS DELIVERY AND ADVANCED DEPLOYMENT TECHNIQUES

Accelerate the delivery of your cloud-native applications with automated delivery, continuous integration/continuous delivery (CI/CD) pipelines, rolling blue/green and canary deployments, and A/B testing.



### STEP 08 EVOLVE A MORE MODULAR ARCHITECTURE

Choose a modular design that makes sense for your specific needs, using microservices, a monolith-first approach, or miniservices—or a combination.

LEARN MORE AT  
[www.redhat.com/en/topics/cloud-native-apps](http://www.redhat.com/en/topics/cloud-native-apps)

READ THE E-BOOK  
*The path to cloud-native applications*  
<https://red.ht/CNADebook>