Architetture Cloud-native

D. Davide Lamanna
CTO, Binario Etico

BinarioEtico

#OSW2021

RIOS RETE ITALIANA OPEN SOURCE

Architetture / Cloud-native

Caratteristiche delle architetture Cloud-native

Sistemi debolmente accoppiati, resilienti, gestibili e osservabili

Cloud-native application come composizione di API-driven-service

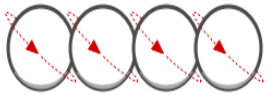
Tecnologie abilitanti:

- Container
- Microservizi
- Service-mesh
- Serverless function
- Immutable infrastructure

Implementate tramite codice dichiarativo

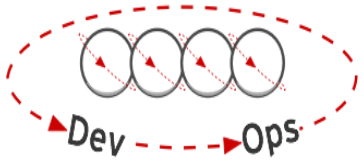
Development Process

Waterfall



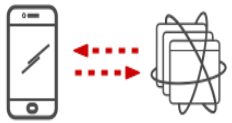
Agile

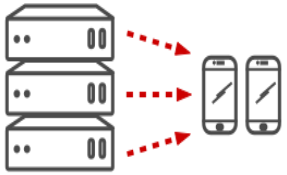
DevOps



Application Architecture

Monolithic





N-Tier

Microservices

Deployment & Packaging

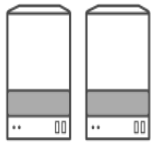
Physical Servers

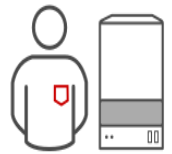
Virtual Servers

Containers

Application Infrastructure

Datacenter





Hosted

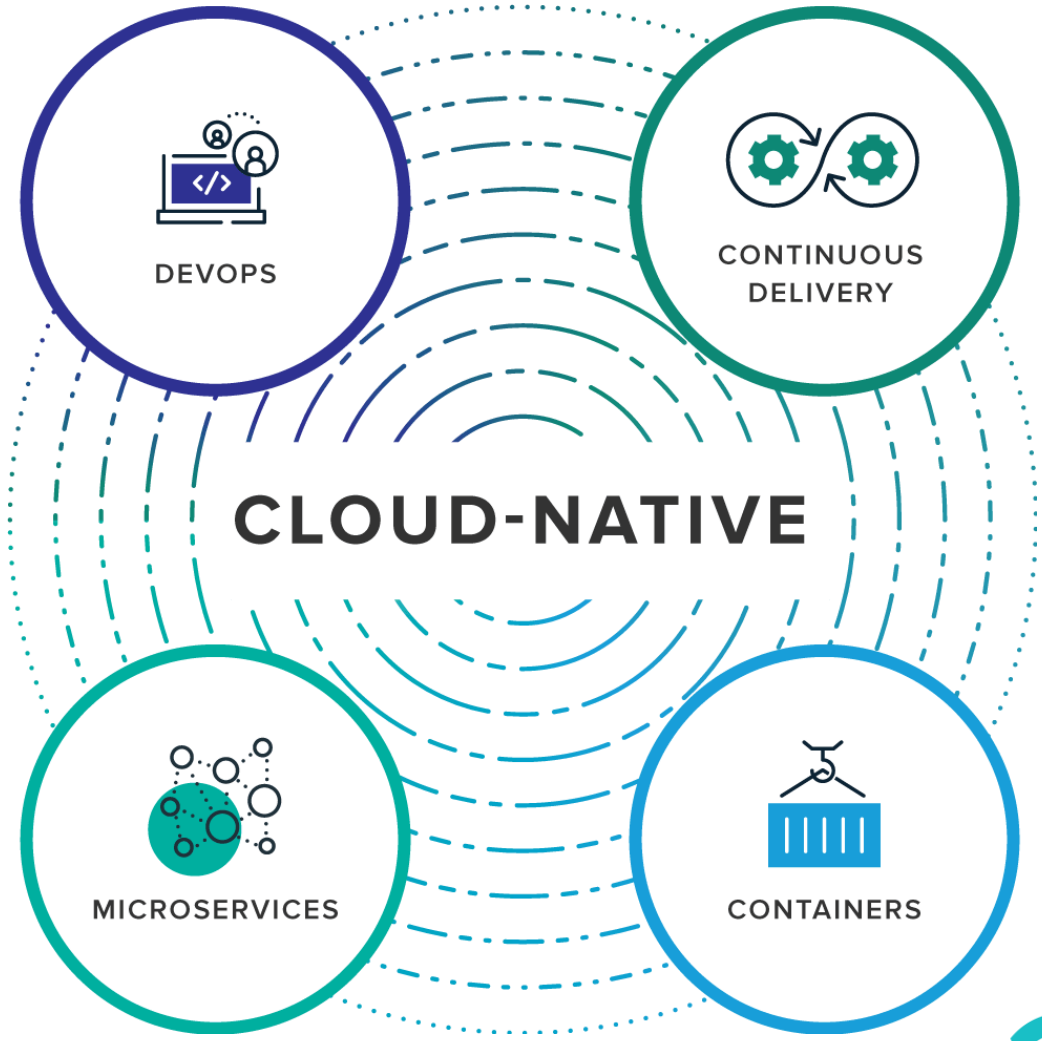
Cloud

OSW2021 OPENSOURCEWEEK

Cloud-native application

Create come un insieme di microservizi che vengono eseguiti in container applicativi e possono essere orchestrate in Kubernetes e gestite e distribuite utilizzando flussi di lavoro DevOps e CI/CD

12(15)-Factor application[1]: metodologia per la creazione di app SaaS che utilizzano formati dichiarativi per l'automazione della configurazione e la portabilità tra ambienti di esecuzione.



[1] Beyond the Twelve-Factor App, by Kevin Hoffman

12(15) Factor Application

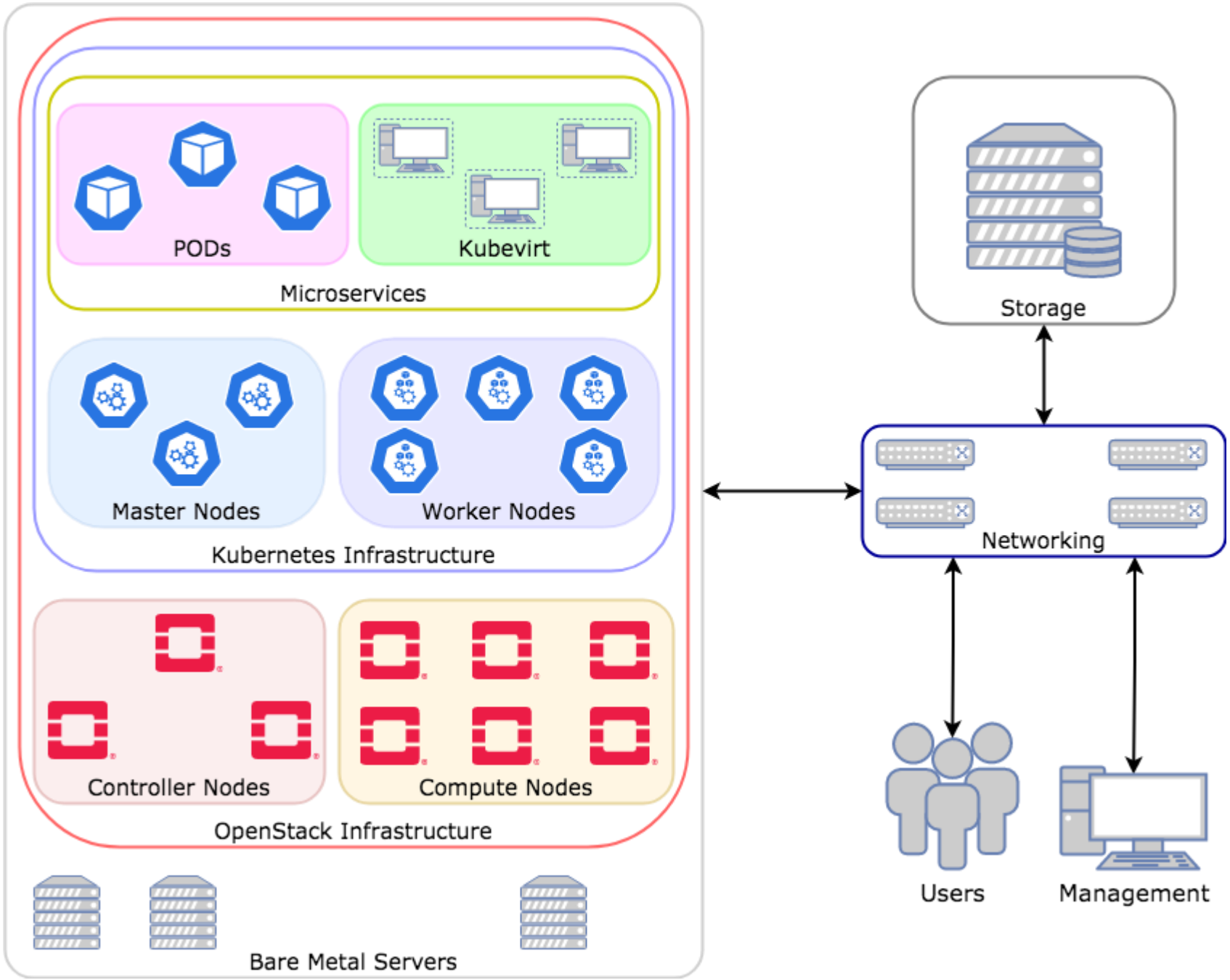
1 - Code Base	Una singola codebase per microservizio, con proprio repository. Versionata, può essere distribuito in più ambienti (qa, staging, produzione).
2 - Dipendenze	Ogni microservizio isola e containerizza le proprie dipendenze, senza influire sull'intero sistema.
3 - Configurazioni	Le informazioni di configurazione al di fuori del microservizio, tramite uno strumento di CM.
4 - Servizi di backup	Database, cache, message broker devono essere esposti tramite un URL (disaccoppiamento dall'applicazione)
5 - Build, versione, esecuzione	Separazione rigorosa tra le fasi di build, rilascio ed esecuzione. Ognuno contrassegnato con un ID univoco, con possibilità di fare rollback.

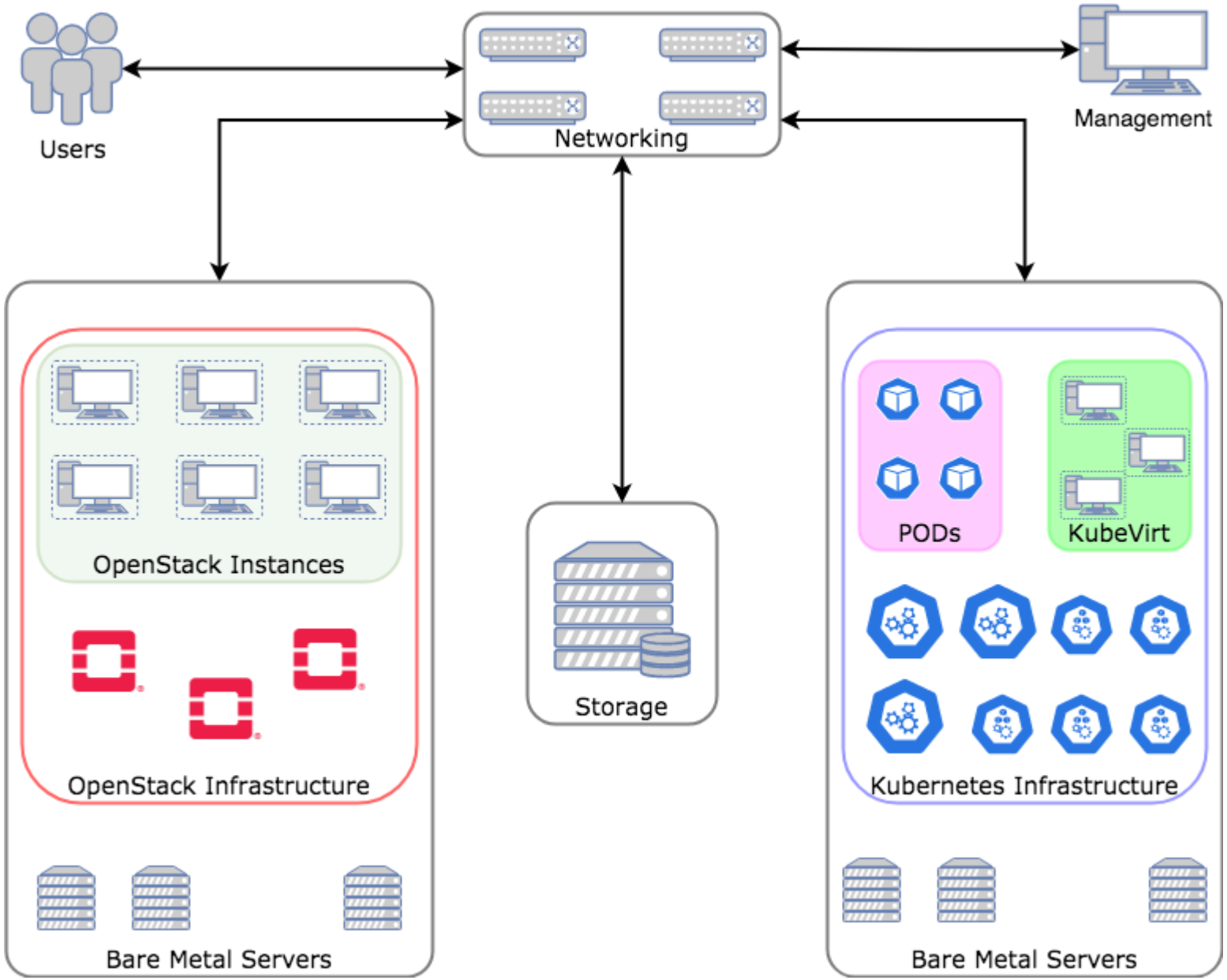
12(15) Factor Application

6 - Processi	Ogni microservizio isolato da altri servizi in esecuzione. Esternalizzare lo stato (cache distribuita o database).
7 - Associazione di porte	Ogni microservizio indipendente, con le interfacce e funzionalità esposte sulla propria porta.
8 - Concorrenza	Quando la capacità deve aumentare, aumentare orizzontalmente i servizi in più processi identici (copie) anziché aumentare una singola istanza di grandi dimensioni.
9 - Disposability	Le istanze del servizio devono essere eliminabili. Favorire l'avvio rapido per aumentare le opportunità di scalabilità e gli arresti per lasciare il sistema in uno stato corretto.
10 - Dev/Prod Parity	Mantenere gli ambienti nel ciclo di vita dell'applicazione il più possibile simili.

12(15) Factor Application

11 - Registrazione	Considerare i log generati dai microservizi come flussi di eventi. Elaborarli con un aggregatore di eventi. Propagare i dati di log a strumenti di data mining/gestione dei log e infine all'archiviazione a lungo termine.
12 - Processi di amministrazione	Eseguire attività amministrative/di gestione, ad esempio la pulizia dei dati o l'analisi dell'elaborazione, come processi una-tantum (strumenti indipendenti, separati dall'applicazione).
13 - API First	Rendere tutto un servizio. Si supponga che il codice verrà utilizzato da un client front-end, un gateway o un altro servizio.
14 - Telemetria	In una workstation si ha una visibilità approfondita sull'applicazione e sul relativo comportamento. Nel cloud non è così. Assicurarsi che la progettazione includa la raccolta di dati di monitoraggio, specifici del dominio e sull'integrità del sistema.
15 - Autenticazione/autorizzazione	Implementare l'identità fin dall'inizio. Si considerino le funzionalità del controllo degli accessi in base al ruolo.


PODs
Kubevirt
Microservices
Master Nodes
Worker Nodes
Kubernetes Infrastructure
Controller Nodes
Compute Nodes
OpenStack Infrastructure
Bare Metal Servers
Storage
Networking
Users
Management



Take away:

1. Container e orchestrazione
2. Conformità
3. Controllo (osservabilità, accessi, percorsi di rete)

Servizi IT affidabili

OSW2021
OPENSOURCEWEEK

BinarioEtico

eLabor
Impariamo lavorando

Geobeyond

ITSERVICE-NET

LARUS

REDTURTLE
agile.open.connected

S2S

SEACOM

SMC

#OSW2021

http://www.reteitalianaopensource.net

RIOS
RETE ITALIANA
OPEN SOURCE