Voci dalla comunità GARRLab

Università degli Studi di Napoli Parthenope

Napoli, 09/11/2021

- Damiano Verzulli
- Francesco Di Nucci
- Giuseppe Masino
- Donato Perruso
- Marco Bauce

WORKSHOP GARR 2021

NET MAKERS

Damiano Verzulli
GARRLab

GARR LAB

Agenda

- GARRLab: recap
- GARRLab@2021:
  - l'approccio cloud-native
  - Studenti alla riscossa
    - **Francesco Di Nucci** (cl. 1998): GitOps, CI/CD, hugo come headless-CMS per il web
    - **Giuseppe Masino** (cl. 2000): Infrastruttura cloud-native: GitOps con Docker, GitLab, Traefik, Portainer e sicurezza all'accesso
    - **Donato Perruso** (cl. 1999): observability (prometheus / grafana), exporters config & deployment
    - **Marco Bauce** (cl. 1999): gestione dichiarativa delle infrastrutture (nix/nixos) e automazione (terraform / morph)

GARRLab: recap

- GARRLab nasce come tecno-community centrata sugli “operativi” dell’ICT negli Atenei: **condivisione della conoscenza** e **crescita professionale** sono i suoi cardini;
- Gruppo Telegram creato l’**11/10/2019**
- **1° anno: +119** partecipanti; **2° anno: +16**
- Grazie a **GARR**, arriva una infrastruttura (su GARR-Cloud) con la quale avviare alcune sperimentazioni

GARRLab: recap

- Come in molte comunità “virtuali”, l’interesse è alto (numerosi iscritti) ma la partecipazione attiva è limitata (gli “attivi” sono meno del 20%)
- La condivisione di conoscenza diventa progressivamente preponderante rispetto alla crescita di competenze

...nel frattempo, li fuori...

- il **Ransomware** fa man bassa;
- il **Cloud-Nazionale** sale agli onori della cronaca;
- la Politica accelera sul problema delle “**competenze ICT**” (...anche nella P.A.);
- alcuni di noi decidono di **cambiare settore**;
- le quote di mercato dei **GAM**, anche nelle NREN, aumentano...

GARRLab: recap

- A 18 mesi dalla partenza, appare chiaro che gli operativi di GARRLab sono programmati per “risolvere le emergenze correnti” e non riescono ad aggredire adeguatamente gli scenari ICT imposti dal mercato (e dall’agenda politica);
- Per mordere i terreni sconfinati delle ~~nuove~~ tecnologie “cloud” e dei ~~nuovi~~ paradigmi applicativi “cloud-native”, a GARRLab servono energie nuove: **STUDENTI**
- Su spunto *reddit* (req info su API di ESSE3), **il 06/03/2021 nasce il gruppo dei GARRistini**

Il gruppo nasce con un obiettivo win-win:

- **agli studenti viene offerta la possibilità (a € 0) di mettere le mani su infrastrutture cloud enterprise (GARR-Cloud) [esperienza, CV] e di confrontarsi con degli “anziani” dell’ICT...**
- **ai GARRisti viene data la possibilità di osservare quello che accade e contribuire a determinarne la linea**

GARRLab@2021: i GARRistini

- L'infrastruttura messa a disposizione degli studenti è stata realizzata ad-hoc, separandola dal resto dell'infrastruttura preesistente

Infra "GARRLab" (preesistente)

LEGACY

Infrastruttura (e servizi) realizzati from-scratch (poche VM; molti container LXC (nelle VM); networking esplicito con OVS; DNS, DHCP, FW, LOG, …, full-managed)

Approccio classico alla HA e al DR (es.: replica postgres fra PA e CT)

Test delle **tecnologie cloud-native** (es.: elasticsearch)

git (GitLab) come supporto alle attività

Infra "GARRistini"

CLOUD Native

- Unica VM (ad-hoc) docker-host, locked; 100% **docker-containers**; admin iniziale via portainer
- Gestione 100% autonoma dei servizi (TLS passtrough su FW-GW)
- **Git (GitLab) come fonte unica della verità**
- Uso estensivo di **CI/CD** (GitLab – approccio GitOps)
- **Scouting tecnologico** (applicativi e tecnologie cloud, automazione, servizi)

GitOps & CI/CD @ GARRLab

Il primo approfondimento sull'architettura allestita con gli studenti ci viene presentato da:

Francesco Di Nucci

dev.francesco.dinucci@protonmail.ch

Corso di Laurea in Ingegneria Meccanica

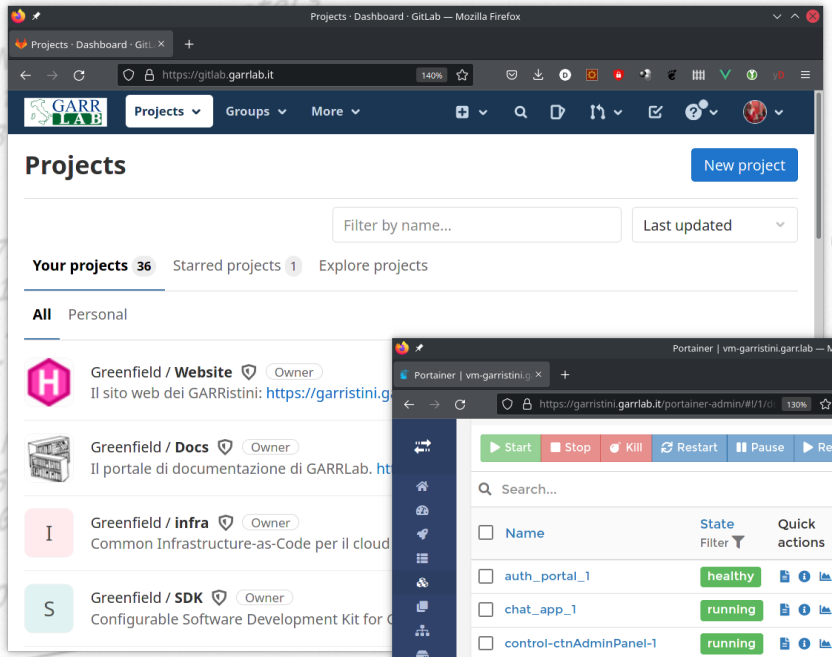
Università degli Studi di Salerno

GitOps & CI/CD @ GARRLab

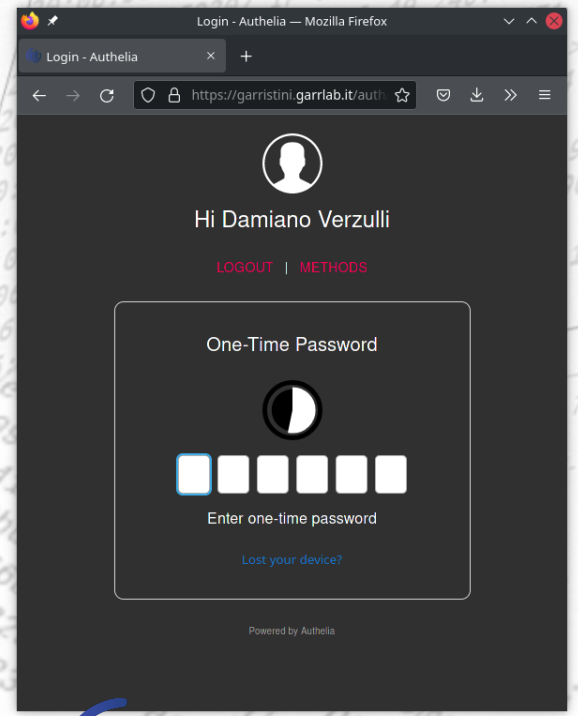
- 100% Self-hosting (GitLab & Authelia)
- “git” come “*unica fonte di verità*” e GITLAB come piattaforma di CI/CD (automazione e riproducibilità)
- Tecnologie a supporto della user-friendliness (Portainer, Hugo, etc.)
- Due esempi:
  - Sito web
  - Bridge Telegram-Matrix

Francesco Di Nucci

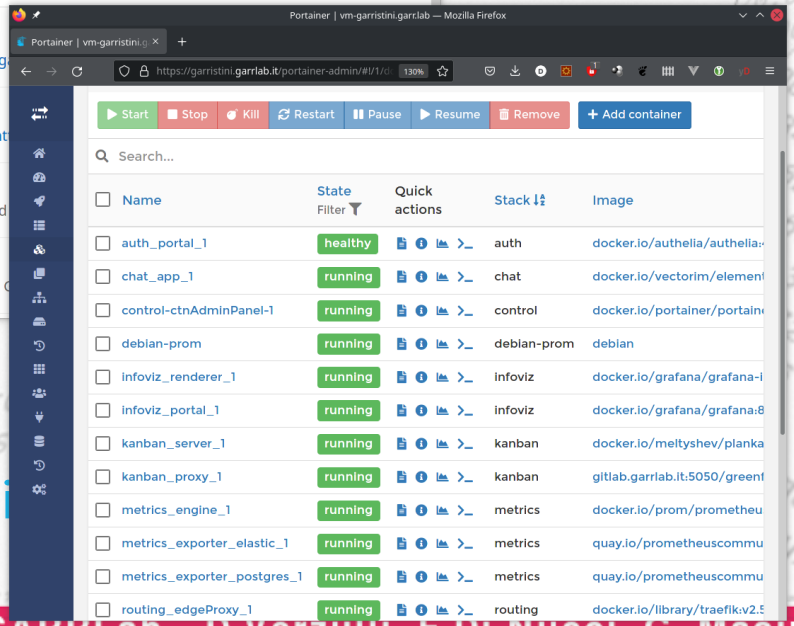
GitOps & CI/CD @ GARRLab



GitLab



Francesco Di Nucci

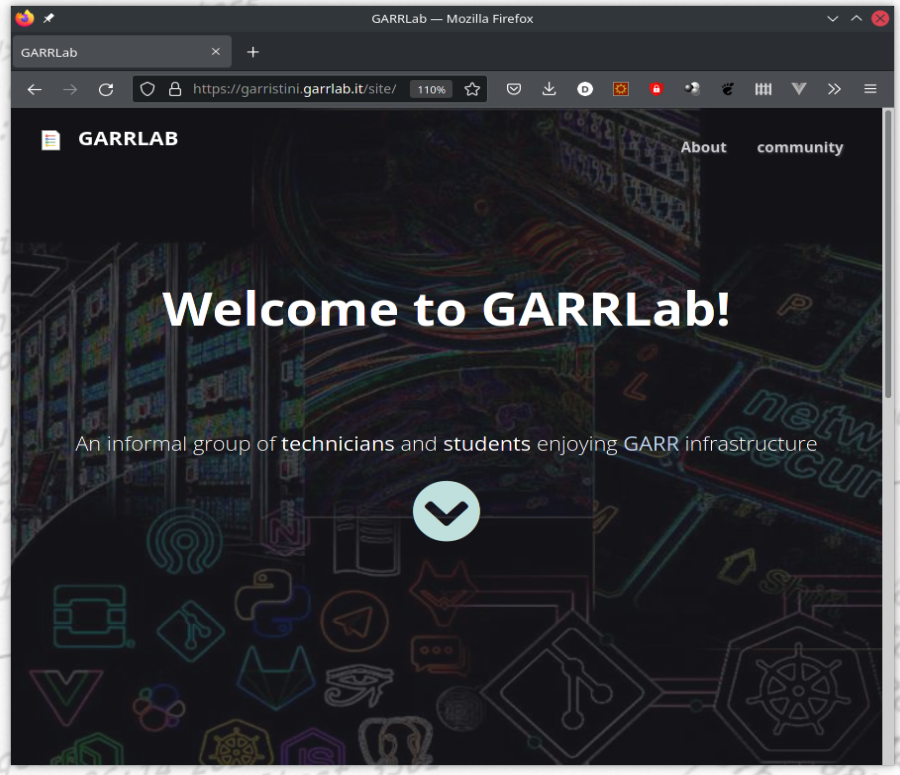


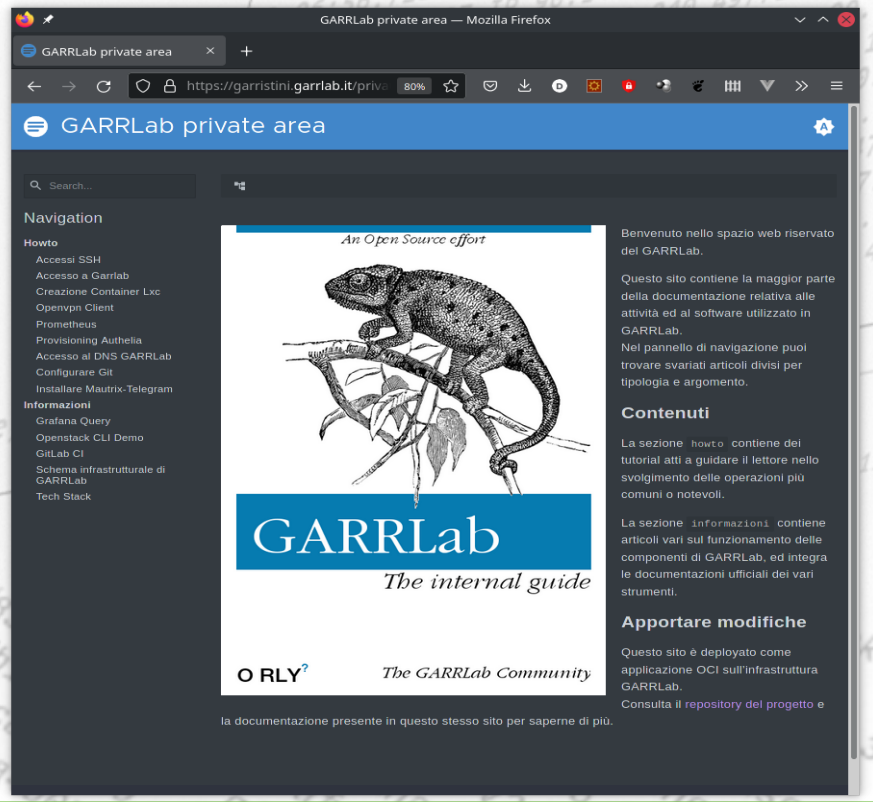
portainer.i

authelia

2FA - TOTP

GitOps & CI/CD @ GARRLab

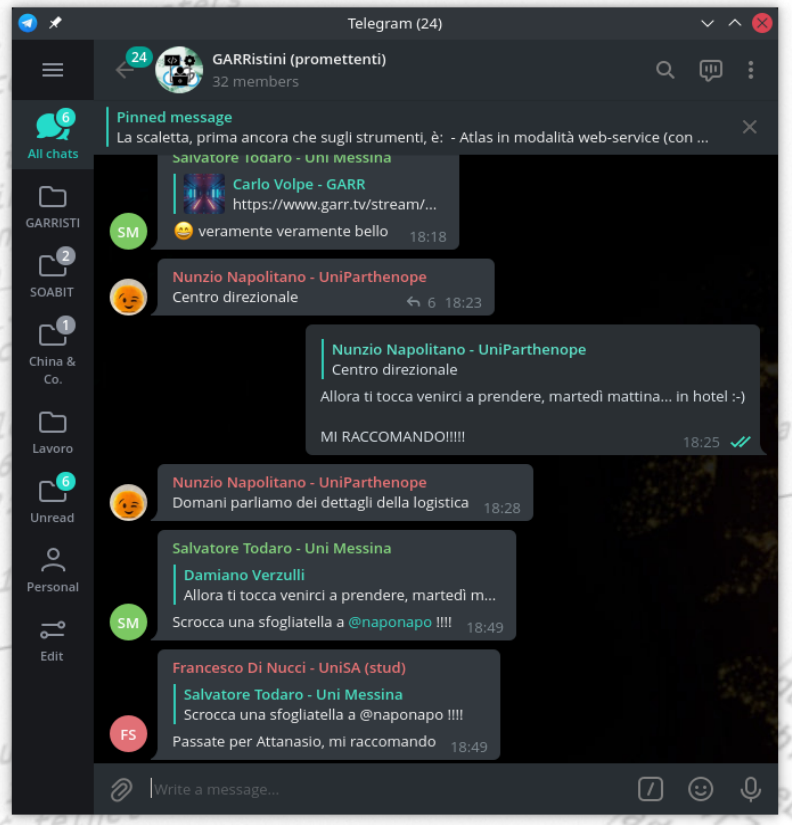


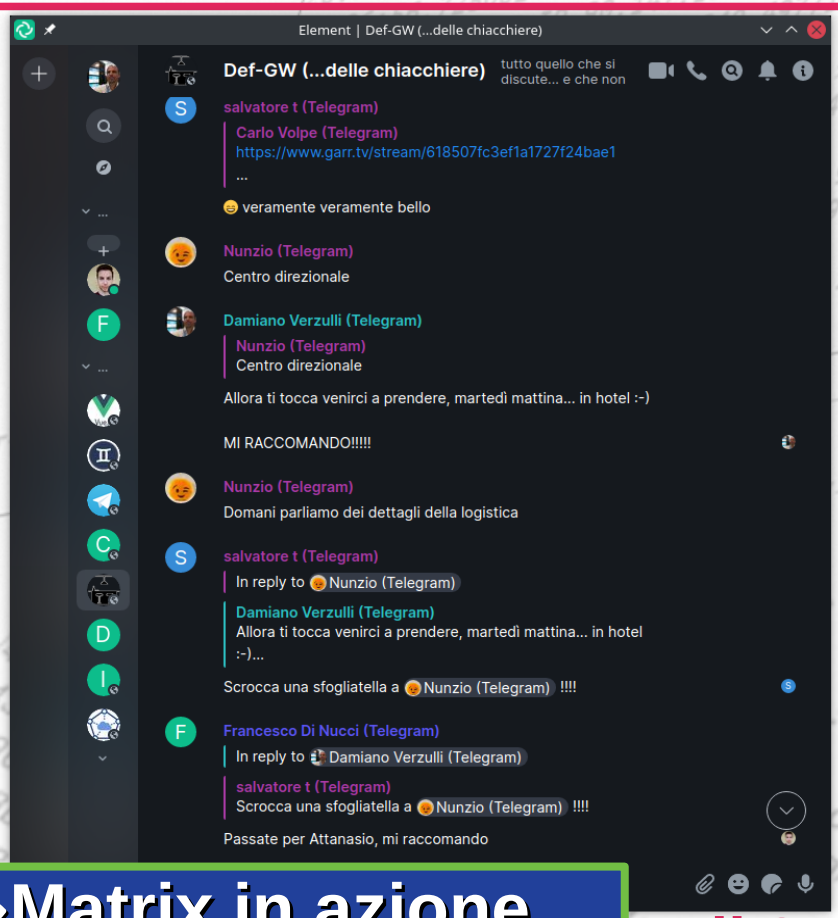


Francesco Di Nucci

Un sito web “pubblico” ed un sito web “privato”, entrambi sviluppati con Hugo e deploiati con l’approccio GitOps & CI/CD

GitOps & CI/CD @ GARRLab





Francesco Di Nucci

Il “bridge” Telegram⇔Matrix in azione

Telegram Desktop (a sinistra); Matrix/Element (a destra)

Infrastruttura cloud-native

Il secondo approfondimento ci viene presentato da:

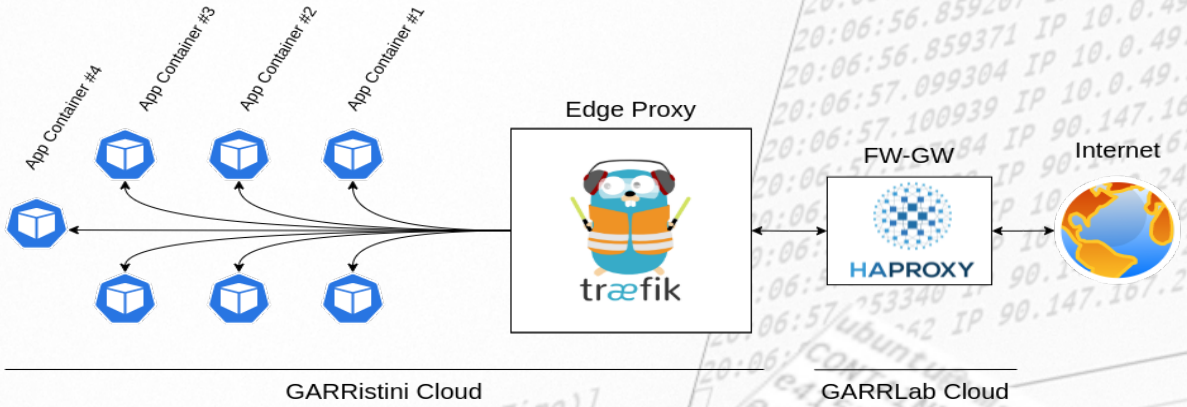
Giuseppe Masino

dev.gmasino@pm.me

Corso di Laurea in Ingegneria Informatica

Università degli Studi di Salerno

Il cloud dei GARRistini



Giuseppe Masino

authelia

Grafana

matrix

portainer.io

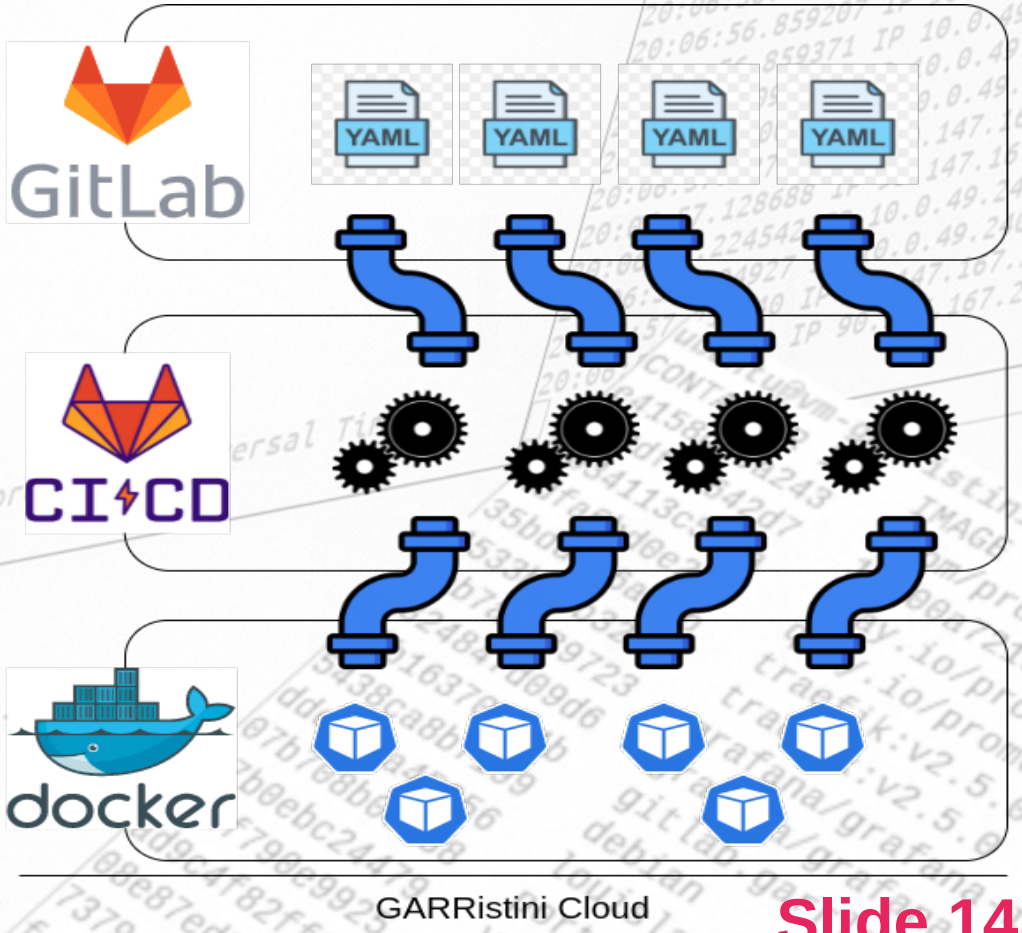
docker

GitLab

Prometheus

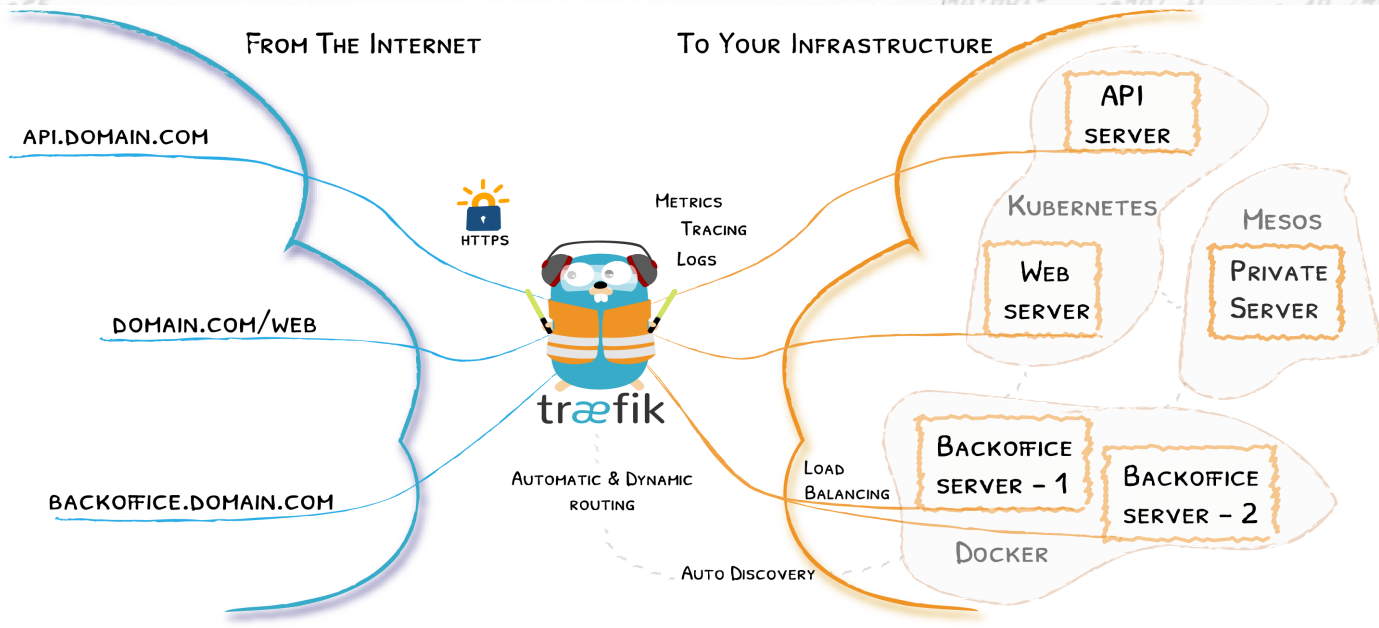
HUGO

Git + CD pipelines + Docker = GitOps



Giuseppe Masino

Traefik: Cloud-native Edge Router



Giuseppe Masino

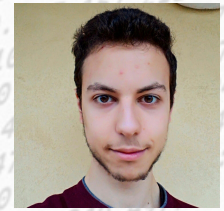
Traefik: Proxy TLS zeroconf *

```yaml
# TLS CertResolvers Config

certificatesresolvers:
  letsencrypt:
    caserver: "https://acme-v02.api.letsencrypt.org/directory"
    tlschallenge: true
    keyType: "EC384"
    storage: "/etc/traefik/acme/acme.json"
    email:   "dev.gmasino+acme@pm.me"


entrypoints:
  web-public:
    address: ":443"
    http:
      tls:
        certresolver: "letsencrypt"
        domains:
        - main: "garristini.garrlab.it"

```

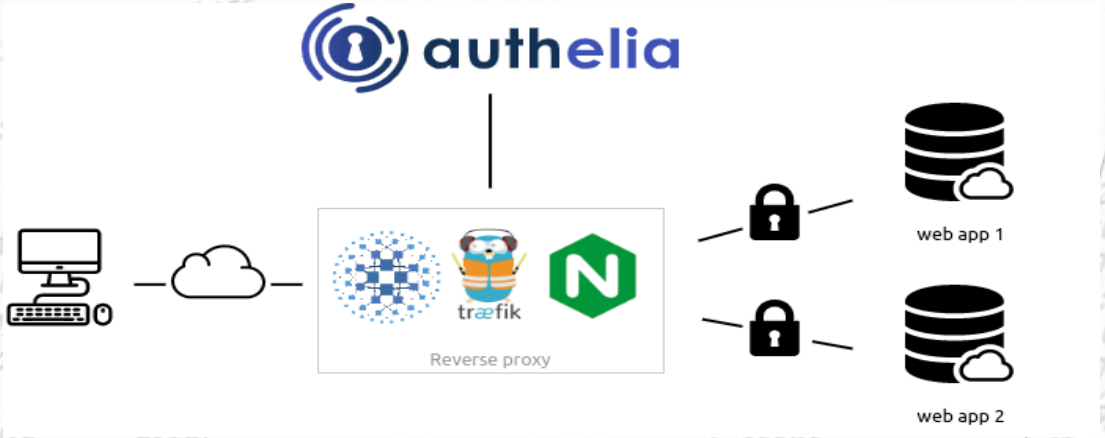


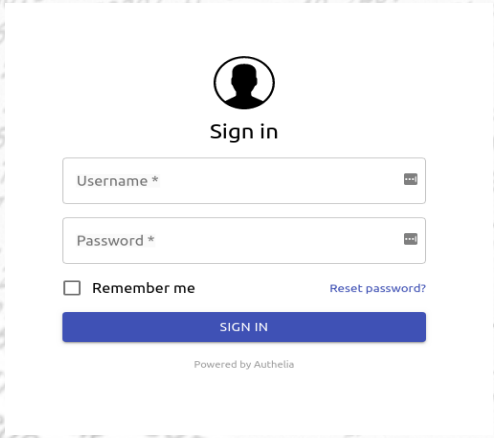
Giuseppe Masino

Dal primo deploy di “produzione”, la gestione dei certificati TLS ha richiesto ZERO intervento umano.

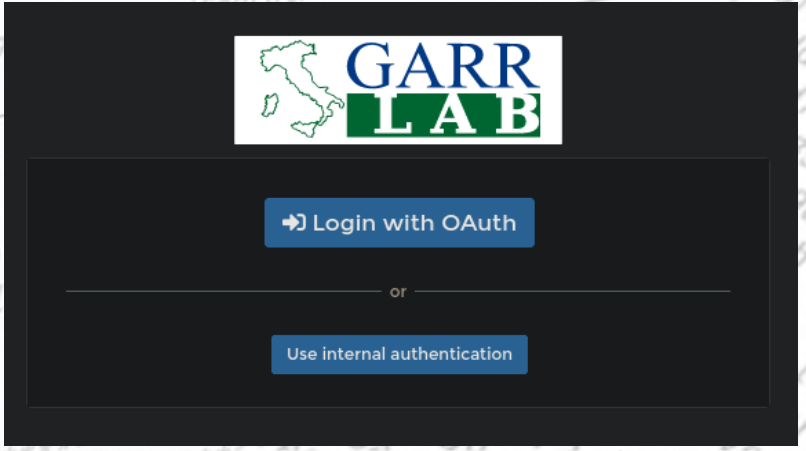
I rinnovi ed i nuovi rilasci avvengono in COMPLETA automazione e con ZERO downtime.

Authentication, Authorization, Accounting





Giuseppe Masino



GitLab

Observability (prometheus / grafana)

Il terzo approfondimento sull'architettura allestita con gli studenti ci viene presentato da:

Donato Perruso

dperruso@pm.me

C.so di Laurea in Ingegneria Elettronica

Università degli Studi di Salerno

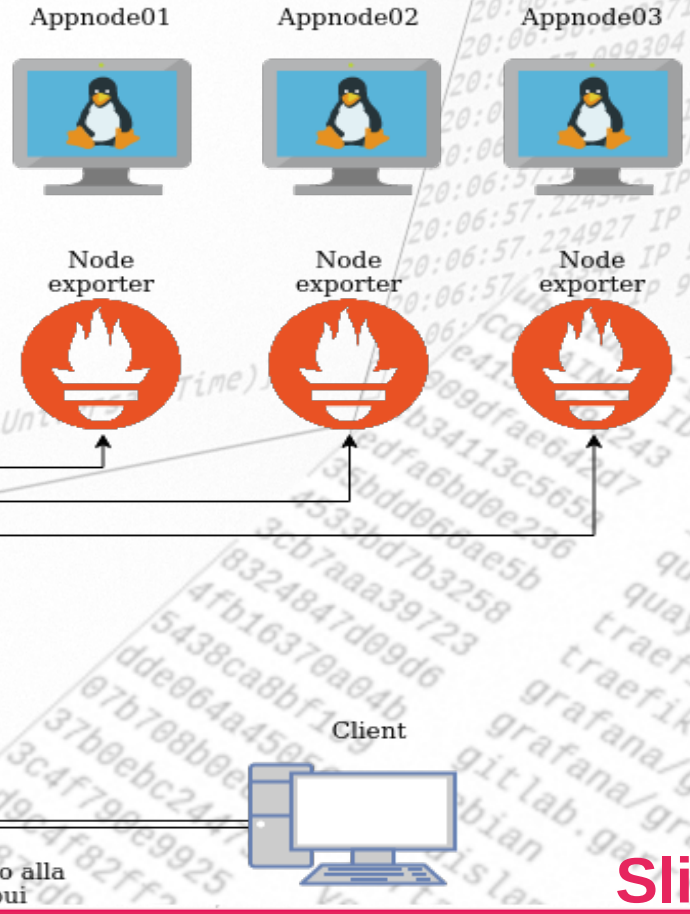
Observability (Prometheus & Grafana)

- Monitoring distribuito utilizzando strumenti e tecnologie cloud-nativeattraverso metriche esportate dalle varie risorse
  - Export di metriche attraverso ***node_exporter***, ***cAdvisor*** ed altri exporters nativi (***Traefik***, ***PowerDNS***)
  - Recupero e archiviazione delle metriche con ***Prometheus***
  - Plotting dei grafici e allestimento dashboard con ***Grafana***

Donato Perruso

Observability (Prometheus & Grafana)

Sistemi trattati:
- 6 Virtual Machine
- 2 Firewall
- 6 Servizi Applicativi

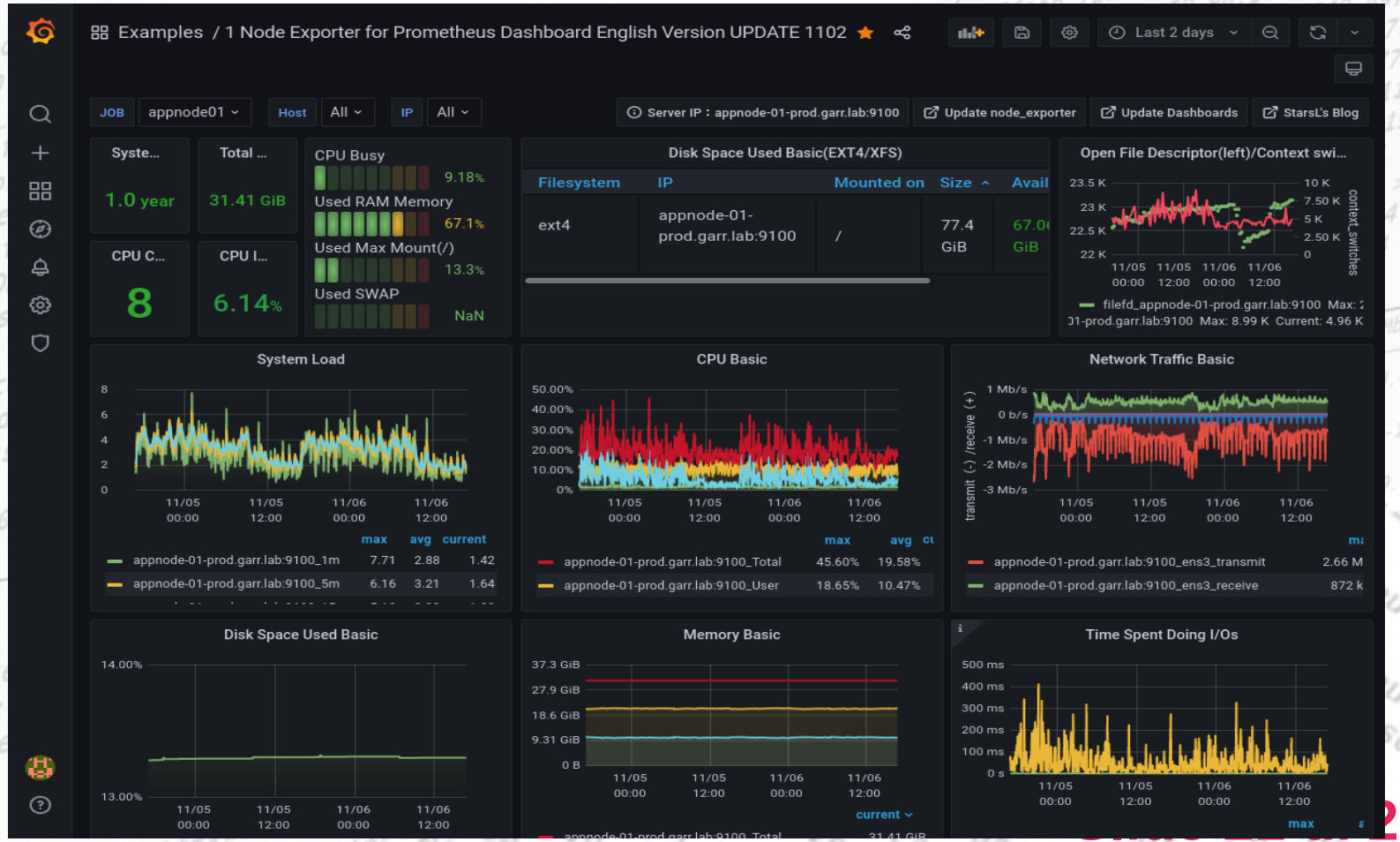


VMs - Nodi

Gli exporter espongono le metriche su webserver ad una specifica porta

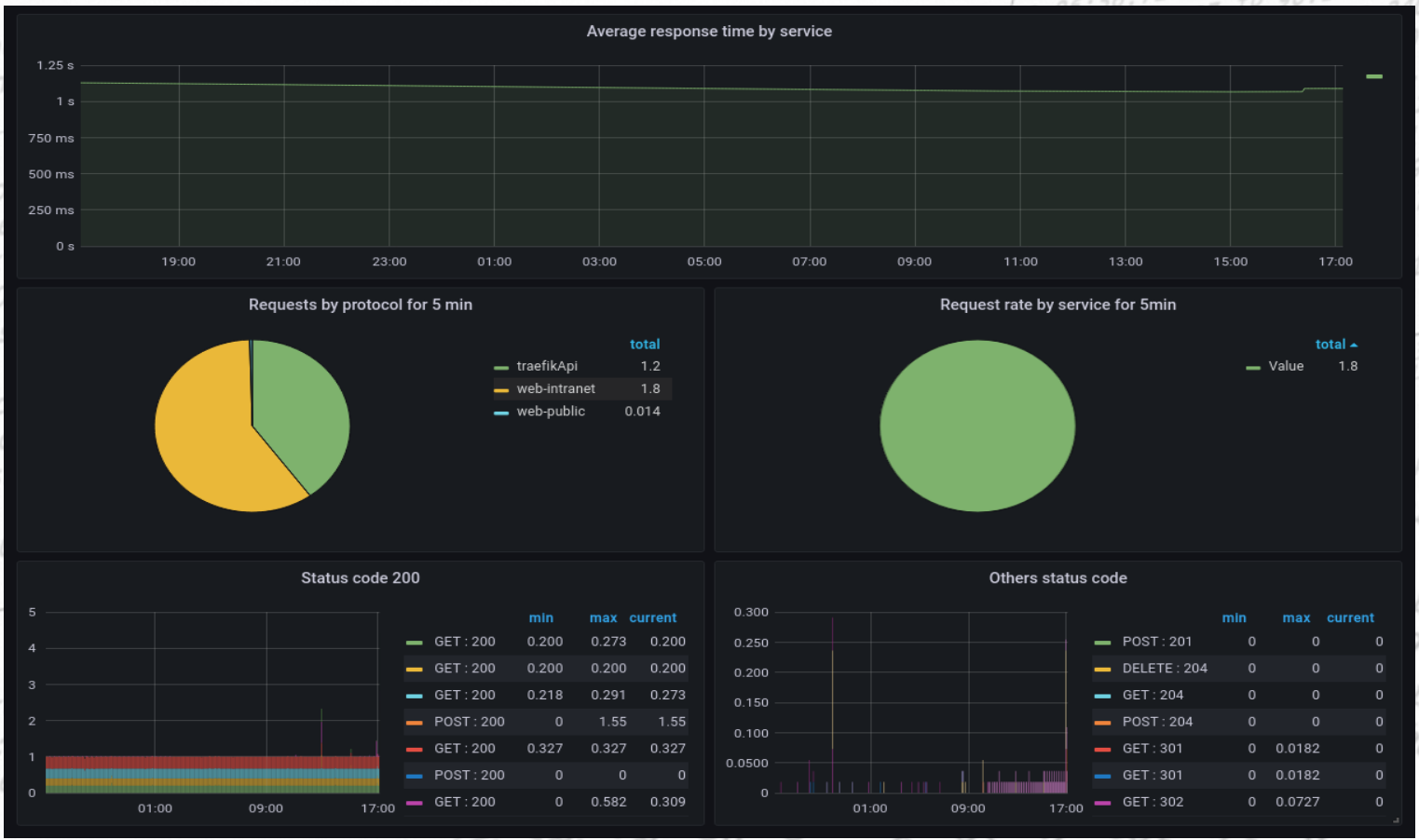
Donato Perruso

Observability (Prometheus & Grafana)



Donato Perruso

Observability (Prometheus & Grafana)



Donato Perruso

Gestione dichiarativa delle infrastrutture

Il quarto approfondimento sull'architettura allestita con gli studenti ci viene presentato da:

Marco Bauce

srht@mrkeebs.eu

C.so di Laurea in Informatica

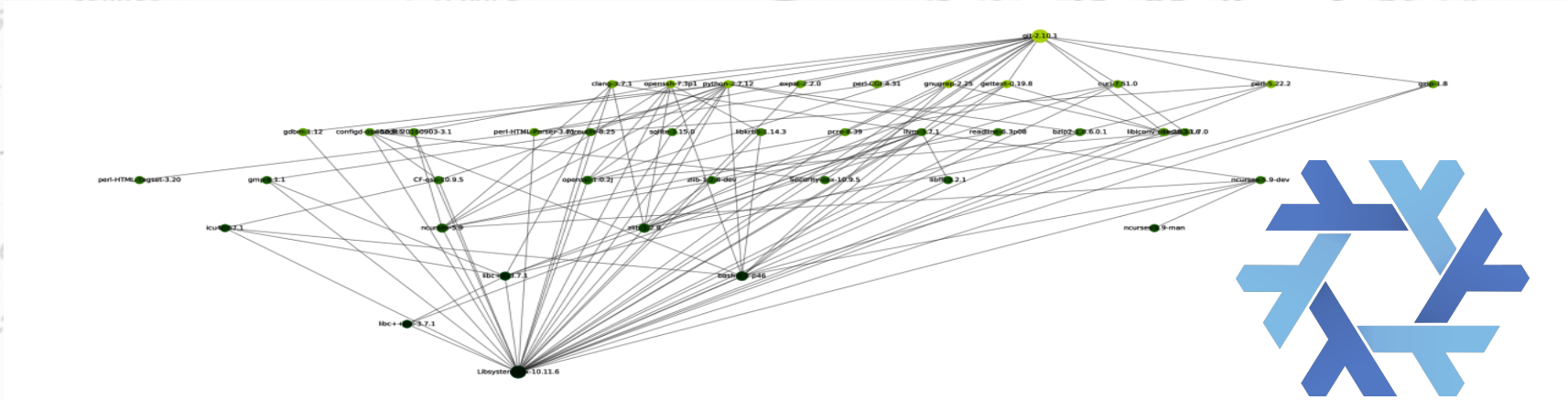
Università degli Studi di Milano

Gestione dichiarativa delle infrastrutture

- Nix Package Manager:
  - Package Manager cross-platform che utilizza un modello di distribuzione puramente funzionale

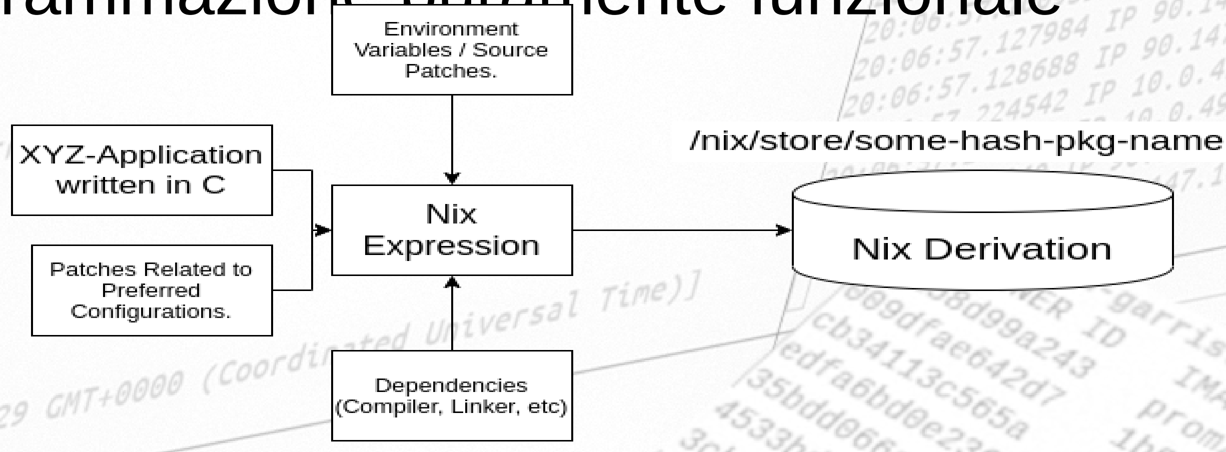
Marco Bauce

es.: nix/store/b6lansfknasfdkaalsdnmf42fas-firefox-31.1



Gestione dichiarativa delle infrastrutture

- Ogni pacchetto è definito grazie ad un linguaggio di programmazione puramente funzionale



Marco Bauce

Questo comporta numerosi vantaggi:

- avere molteplici versioni del singolo pacchetto.
- assicura che tutte le dipendenze siano installate.
- ogni pacchetto e' atomico e questo permette un rollback
- gestione il piu deterministica possibile

Gestione dichiarativa delle infrastrutture

NixOS

- **NixOs**: Estensione di Nix all'intero sistema operativo
  - Riproducibilità totale
  - Dichiaratività

Alcuni strumenti per la gestione di una pluralità di hosts NixOs:

- NixOps
- Morph
- Colemana

```
{ config, pkgs, lib, ... }:

{
  imports = [ ./foo.nix ./bar.nix ];

  boot = {
    loader = {
      efi.efiSysMountPoint = "/boot/EFI";
    };
    initrd = {
      kernelModules = [ "amdgpu" ];
      luks.devices = {
        ...;
      };
    };
  };

  networking = {
    hostname = "test";
    wireless.enable = true;
    firewall = {
      enable = true;
      allowedTCPPorts = [ 80 ];
    };
  };

  environment.systemPackages = with pkgs; [
    emacs firefox vim
  ];

  services = {
    openssh = {
      enable = true;
    };

    openvpn.servers = {
      garrlabVPN = { ... };
    };

    xserver = {
      enable = true;
      desktopManager.plasma5 = {
        enable = true;
      };
    };
  };
}
```

Marco Bauce

Gestione dichiarativa delle infrastrutture

- **Terraform** per gestire e creare le risorse
- Uso del JSON creato da terraform per gestire le risorse create da Morph

HashiCorp Terraform

Marco Bauce

```
terraform {
  required_providers {
    lxd = {
      source = "terraform-lxd/lxd"
    }
  }
}

provider "lxd" {
  generate_client_certificates = false
  accept_remote_certificate = true

  lxd_remote {
    name = "ubuntu-garr"
    address = "172.16.17.130"
    scheme = "https"
    default = true
  }
}

resource "lxd_container" "test1" {
  name      = "test1"
  image     = "nixos"
  ephemeral = false

  config = {
    "boot.autostart" = true
  }

  limits = {
    cpu = 2
  }
}

resource "lxd_container" "test2" {
  name      = "test2"
  image     = "nixos"
  ephemeral = false

  config = {
    "boot.autostart" = true
  }

  limits = {
    cpu = 4
  }
}
```

Provider LXD

2 x Container NixOs (LXD)

TERRAFORM

```
let
  resourcesByType = (import ./parsetf.nix { }).resourcesByType;
  lxd_container = resourcesByType "lxd_container";

  test1 = resource: { config, pkgs, modulesPath, lib, name, ... }: {
    imports =
      [ ./common.nix
      ];

    services = {
      postgresql.enable = true;
      postgresql.package = pkgs.postgresql_11;
    };

    deployment.targetHost = resource.values.ip_address;
    networking = {
      hostName = resource.name;
    };
  };

  test2 = resource: { config, pkgs, modulesPath, lib, name, ... }: {
    imports =
      [ ./common.nix
      ];

    services.nginx = {
      enable = true;
      virtualHosts.default = {
        default = true;
        locations."/".return = "200 \"Hello from ${name} at ${resource.values.ip_addr
ess}\"";
      };
    };

    deployment.targetHost = resource.values.ip_address;
    networking = {
      hostName = resource.name;
      firewall.allowedTCPPorts = [ 80 ];
    };
  };


in
{
  network = {
    pkgs = import
      (builtins.fetchGit {
```

Tweak

Container gestiti

MORPH