Raspberry Pi Cluster – Progettazione Dettagliata e Appunti

Questo documento approfondisce in particolare **le fasi iniziali di progettazione e valutazione**, prima ancora di scegliere hardware e software specifici.

---

1. Visione, contesto e casi d'uso

1.1 Scopo principale del cluster

Prima di tutto, definisci chiaramente **perché** vuoi il cluster. Ad esempio:

- Eseguire bot (Telegram, Discord, automazioni varie).
- Ospitare un sito web personale o più siti.
- Fornire funzione di NAS (condivisione file, backup, archivio).
- Ambiente di laboratorio / sperimentazione (test nuove tecnologie, microservizi, ecc.).

Scrivi una frase sintetica di scopo, ad esempio:

> "Voglio un cluster di Raspberry Pi che mi permetta di eseguire bot e un sito personale 24/7, con spazio per un NAS casalingo e spazio per sperimentare nuove applicazioni."

Questa frase ti servirà per fare scelte coerenti (se una scelta non supporta questa frase, probabilmente è sbagliata o eccessiva).

1.2 Tipologia di utilizzo

Per ogni servizio che pensi di ospitare, definisci:

- **Frequenza d'uso** (sempre, giornaliero, saltuariamente).
- **Interattività** (in tempo reale / batch / asincrono).
- **Client principali** (tu solo, pochi utenti fidati, internet aperto).

Esempio tabella di lavoro (da compilare personalmente):

Servizio	Utenti	Uso	Criticità percepita
Bot Telegram X	Solo io	Sempre on	Media
Sito personale	Pubblico web	Sempre on	Bassa/Media
NAS documenti	Io + famiglia	Sporadico	Alta
Lab test	Solo io	Sporadico	Bassa

Questa prima classificazione ti servirà per legare ogni servizio a esigenze di **affidabilità, storage, backup e sicurezza**.

---

2. Requisiti funzionali e non funzionali

2.1 Requisiti funzionali (cosa deve poter fare il sistema)

Per ogni servizio, rispondi a domande come:

- Il bot deve poter:
  - rispondere in pochi secondi ai messaggi?
  - accedere a determinati file / API interne?
- Il sito web deve:
  - servire solo pagine statiche?
  - avere backend dinamico (database, login)?
- Il NAS deve:
  - permettere condivisione file in LAN soltanto?
  - essere accessibile anche da remoto?
  - supportare più utenti con permessi diversi?

Scrivi una lista puntata di azioni che **devono** essere possibili. Esempio:

- Bot:
  - rispondere ai comandi in meno di 5 secondi;
  - poter leggere e scrivere su un volume persistente;
  - poter riavviarsi automaticamente se va in crash.
- Sito:
  - accessibile via HTTPS da internet;
  - log di accesso salvati almeno 30 giorni.
- NAS:
  - una share privata (solo tua) + eventuale share comune;
  - accesso da LAN + (forse) accesso da remoto via VPN.

2.2 Requisiti non funzionali

Qui definisci **quanto bene** deve funzionare il sistema, non solo **cosa** fa.

Disponibilità / Uptime Per ogni servizio, decidi un obiettivo realistico, ad esempio:

- Bot personale: downtime di qualche ora ogni tanto accettabile.
- NAS documenti importanti: vorresti che fosse quasi sempre disponibile, ma è accettabile un downtime di qualche ora se puoi comunque recuperare i dati dai backup.
- Sito personale: se è offline qualche ora non è un dramma.

Definisci in forma esplicita:

- **Accettazione downtime**:
  - "Posso tollerare fino a X ore di downtime al mese per questo servizio."

RTO e RPO (anche solo qualitativi)

- **RTO (Recovery Time Objective)**: quanto tempo è accettabile per ripristinare il servizio dopo un guasto.
  - Es. "Per il NAS, un RTO di 24 ore è accettabile".
- **RPO (Recovery Point Objective)**: quanti dati puoi permetterti di perdere in caso di disastro (quanto sono vecchi i backup accettabili).
  - Es. "Per i documenti importanti sul NAS, RPO massimo 24 ore (backup giornaliero)".
  - "Per il lab test, RPO indifferente (posso perdere tutto)."

Prestazioni

- Stima molto grossolana, ma utile:
  - Numero di utenti contemporanei del sito (1–10, 10–100, …).
  - Dimensione media dei file nel NAS (KB, MB, video da GB).
  - Traffico previsto (es. streaming video via NAS o solo documenti?).

Queste valutazioni indicano se ti basta un singolo disco + Gigabit, o se ha senso pensare a soluzioni più performanti (spesso su Raspberry non serve esagerare).

Sicurezza Decidi livelli di sicurezza per categoria di servizi/dati:

- Dati "pubblici" (sito statico): confidenzialità bassa, integrità importante.
- Dati personali sul NAS: confidenzialità alta.
- Accesso amministrativo (SSH, UI gestione): confidenzialità molto alta.

Stabilisci almeno:

- "Tutto ciò che è amministrazione (SSH, UI cluster, NAS) è accessibile solo tramite VPN o rete fidata, non direttamente esposto su internet."
- "I servizi pubblici esposti (es. sito) passano da un unico reverse proxy con HTTPS."

---

3. Vincoli, assunzioni e contesto reale

Prima di scegliere tecnologia, devi chiarire i vincoli pratici.

3.1 Vincoli fisici

- Dove sarà fisicamente il cluster?
  - stessa stanza, casa diversa, ufficio?
- Come sono:
  - la ventilazione e la temperatura ambiente?
  - il rumore accettabile (ventole)?
- Hai uno spazio dedicato (shelf, armadio, rack) o sarà su una scrivania?

Questo influenzerà:

- tipo di case;
- necessità di ventole/raffreddamento attivo;
- uso o meno di PoE (meno cavi di alimentazione).

3.2 Energia e alimentazione

- Il cluster sarà sempre acceso?
- Hai prese limitate o no?
- Vuoi un UPS? Di che taglia?

Stima molto grossolana di consumo:

- Raspberry Pi 4/5: indicativamente 3–7 W a seconda del carico (più per dischi esterni).
- 3–4 nodi + dischi: ti muovi facilmente tra 20–40 W complessivi.

Questo ti permette di capire:

- costo elettrico indicativo al mese;
- capacità minima dell'UPS (almeno per chiudere in sicurezza in caso di blackout).

3.3 Vincoli di rete

- Il router che userà il cluster lo gestisci tu?
  - puoi configurare port forwarding?
  - supporta DDNS?
- Hai un IP pubblico o sei dietro CGNAT (NAT del provider)?
- La connettività cambia spesso (es. lo sposti in posti diversi)?

Se:

- hai IP pubblico e router configurabile → puoi usare DDNS + port forwarding se il cluster resta fisso;
- sei dietro CGNAT o vuoi che il cluster sia "mobile" → meglio VPS + VPN o tunneling (Cloudflare, Tailscale, ecc.).

3.4 Budget e orizzonte temporale

- Budget iniziale indicativo (solo per l'hardware): X €.
- Quanto spesso sei disposto a investire di nuovo (es. ogni 1–2 anni)?
- Quanto tempo pensi di dedicare alla gestione:
  - 1–2 ore a settimana? meno? di più?

Un cluster "ricco" di funzionalità (orchestratore, storage distribuito, HA) richiede più **tempo mentale** che soldi; è importante essere realistici su quanto tempo avrai voglia di dedicarci.

---

4. Scelte architetturali di alto livello

Una volta chiariti obiettivi, requisiti e vincoli, passi alle **scelte macro**.

4.1 Single node "grosso" vs cluster di nodi piccoli

Valuta:

- Ti serve davvero un cluster, o ti basterebbe un Raspberry "potente" + buon storage e backup?
- Il vantaggio del cluster sta in:
  - separazione dei ruoli (NAS, compute, test);
  - resilienza parziale (un nodo guasto non ferma tutto);
  - possibilità di spostare carico tra nodi;
  - ambiente di studio per tecnologie "da cluster" (Kubernetes, ecc.).

Criterio pratico:

- Se il tuo obiettivo principale è **NAS + pochi servizi** → può bastare un nodo singolo ben fatto + backup seri.
- Se vuoi anche **imparare/giocare con concetti di cluster** → ha senso 3+ nodi.

4.2 Orchestratore o gestione manuale?

Opzioni principali:

1. **Gestione manuale (systemd + docker singolo-nodo)**
   - Pro:
     - semplice;
     - meno componenti da imparare.
   - Contro:
     - devi gestire tu la distribuzione dei servizi sui nodi;
     - meno trasparenza: sei più consapevole di "dove gira cosa".
2. **Docker Swarm**
   - Pro:
     - relativamente semplice rispetto a Kubernetes;
     - concetto di servizi, stack, replica;
     - integrazione naturale con Docker.
   - Contro:
     - meno ecosistema rispetto a K8s;
     - progetto meno "di moda", ma ancora valido per home lab.
3. **k3s (Kubernetes leggero)**
   - Pro:
     - experience più vicina a Kubernetes completo;
     - scheduling automatico, servizi, ingress, ecc.;
     - ecosistema ricco (helm, operator, ecc.).
   - Contro:
     - curva di apprendimento più ripida;

  - più moving parts da gestire.

Criterio:

- Se vuoi **massima trasparenza dell'insieme di nodi come "un cluster" di servizi**, e sei disposto a studiare un po', k3s o almeno Swarm sono fortemente consigliati.
- Se vuoi solo qualche servizio e non ti interessa troppo il "cluster vero", gestione manuale può bastare.

4.3 Architettura di storage

- **Centralizzato**:
  - Un nodo NAS con dischi collegati;
  - altri nodi montano via NFS/SMB (per dati) o i servizi vi accedono tramite rete.
  - Più semplice mentalmente, ma il NAS diventa un punto critico.
- **Distribuito**:
  - Vari nodi con dischi e un layer software sopra (es. GlusterFS, Ceph, ecc.).
  - Permette ridondanza e distribuzione dei dati;
  - molto più complesso, spesso eccessivo per un homelab.

Criterio pratico per iniziare:

- Parti con **storage centralizzato + buon piano di backup**.
- Se in futuro ti sembrerà necessario, potrai valutare storage distribuito.

4.4 Accesso remoto e IP "statico"

Decidi in modo esplicito quale modello adottare:

- **Modello 1 – Casa fissa con IP pubblico**
  - DDNS + port forwarding;
  - reverse proxy sul nodo front-end;
  - (idealmente) VPN per accesso amministrativo.
- **Modello 2 – Cluster mobile / reti variabili**
  - VPS con WireGuard come "hub";
  - i Raspberry si collegano alla VPS;
  - reverse proxy sulla VPS che punta ai servizi nel cluster via VPN.
  - Ipotesi alternativa: Cloudflare Tunnel / Tailscale per bypassare problemi di NAT.

Questa scelta è fondamentale: determina come penserai a DNS, certificati HTTPS, sicurezza, ecc.

---

5. Modello di trasparenza della molteplicità dei nodi

Qui si collega la parte iniziale di architettura con il desiderio di usare i Raspberry "come fossero uno solo".

5.1 Livelli di trasparenza realistici

- **Trasparenza lato servizi**:
  - un solo hostname/IP per chi si collega;
  - i servizi sono schedulati su nodi diversi senza che tu debba decidere ogni volta.
- **Trasparenza lato amministrazione**:
  - un solo punto di accesso SSH;
  - un'unica UI web di gestione (es. Portainer, dashboard K8s).
- **Non trasparente lato hardware**:
  - a livello kernel/desktop/risorse fisiche i nodi restano separati;
  - un cavo HDMI mostra solo un nodo, non la somma dell'hardware.

5.2 Nodo master/bastion + worker

Definisci fin dall'inizio la topologia logica:

- 1 nodo **master/bastion**:
  - hostname: es. `cluster-master`;
  - punto unico di accesso SSH (anche via VPN);
  - eventualmente con interfaccia grafica collegata a monitor HDMI;
  - ospita le UI di gestione (Portainer, dashboard, ecc.).
- N nodi **worker** (es. `rpi-1`, `rpi-2`, `rpi-3`):
  - senza schermo;
  - collegati solo via rete e orchestratore.

Da documentare:

- mappa IP/hostname di tutti i nodi;
- ruoli principali (master/worker/NAS).

5.3 Modello di gestione via SSH

Fin da subito, pianifica e scrivi:

- "Mi collegherò sempre al nodo master (`cluster.miodominio.it`) per amministrare il cluster."
- "I worker non saranno raggiungibili direttamente da internet."

Usa un file di configurazione SSH (sul tuo PC/VM di amministrazione) per rendere trasparente l'accesso ai nodi dietro al master, tramite `ProxyJump`.

In questo modo:

- a livello operativo, non ti "accorgi" di dover passare per il master;

- i comandi e gli script possono usare nomi logici (`rpi-1`, `rpi-2`, ecc.) senza conoscenza del routing.

5.4 Modello per HDMI e UI dell'OS

Definisci:

- quale Raspberry sarà il **desktop del cluster**:
  - Raspberry Pi OS Desktop;
  - monitor HDMI, tastiera, mouse;
  - software client (browser, terminali, editor).

Questo nodo:

- è il tuo "PC" locale dentro il cluster;
- da qui accedi:
  - alle UI web (Portainer, dashboard);
  - via SSH agli altri nodi;
  - a eventuali desktop remoti degli altri nodi (via VNC/X11).

È importante accettare che:

- non avrai un singolo desktop "fuso" con le risorse di tutti i Raspberry;
- ma hai un solo punto grafico da cui **controlli il cluster**.

5.5 VMware / VM come punto di controllo esterno

Se hai un PC o un server con VMware (Workstation / Player / ESXi), decidi se vuoi:

- creare una VM Linux dedicata come **stazione di amministrazione**:
  - da lì fai SSH verso il master;
  - apri le UI web in browser;
  - gestisci file di configurazione, script e IaC (Ansible, Terraform ecc.).

Dal punto di vista progettuale, questa VM è la tua "console" remota del cluster.

---

6. Processo iniziale di progettazione passo per passo

Riassunto operativo delle fasi iniziali, con più dettaglio:

1. **Definisci la vision in 1–2 frasi**
   - Cosa vuoi che il cluster faccia, per chi, e quanto deve essere affidabile.
2. **Fai un elenco dei servizi previsti**
   - Bot, sito, NAS, lab, altro;
   - per ciascuno: utenti, frequenza d'uso, criticità, accesso (LAN/Internet).
3. **Attribuisci requisiti non funzionali per servizio**
   - Uptime desiderato (anche grossolano);
   - RTO/RPO qualitativi;

- sensibilità dei dati (bassa/media/alta);
- prestazioni richieste (bassa/media/alta).

4. **Raccogli vincoli realistici**
   - Fisici (spazio, temperatura, rumore);
   - di rete (IP pubblico, port forwarding, CGNAT);
   - economici (budget iniziale);
   - di tempo (quante ore/mese puoi dedicare alla gestione).
5. **Scegli il modello di accesso remoto**
   - Casa fissa: DDNS + port forwarding + eventualmente VPN interna;
   - Cluster "mobile": VPS + WireGuard oppure Cloudflare Tunnel/Tailscale.
6. **Decidi se ti serve un cluster vero o un nodo singolo**
   - Se vuoi imparare concetti di cluster e distribuire servizi → 3+ nodi;
   - se ti basta un server casalingo unico → un Pi potente con storage serio + backup.
7. **Scegli orchestrazione (se fai cluster)**
   - Obiettivo di semplicità → Docker Swarm;
   - Obiettivo di vicinanza al mondo enterprise → k3s/Kubernetes.
8. **Decidi topologia ruoli dei nodi**
   - 1 master/bastion (con o senza desktop);
   - X worker;
   - eventualmente 1 nodo NAS (che può coincidere con master o worker).
9. **Definisci la mappa IP/hostname**
   - Sottorete dedicata (es. 192.168.10.0/24);
   - IP statici o prenotazioni DHCP per ogni nodo;
   - nomi coerenti (`cluster-master`, `rpi-1`, `rpi-2`, `nas-node`, ecc.).
10. **Progetta la gerarchia di storage**
    - Dove risiede il sistema operativo (SSD su ogni nodo);
    - dove risiedono i dati del NAS;
    - dove risiedono i volumi persistenti dei container;
    - strategia di backup (dischi esterni, altro NAS, cloud, VPS).
11. **Disegna uno schema logico (anche su carta)**
    - Nodi con i ruoli;
    - connessioni di rete (LAN, VPN, internet);
    - flusso di accesso (utenti → reverse proxy/VPS → servizi nel cluster);
    - punti di autenticazione (VPN, SSH, UI web).

Dopo queste fasi, sei pronto a:

- dimensionare l'hardware (numero di Pi, RAM per nodo, capacità dei dischi, switch, UPS);
- scegliere le tecnologie concrete (OS, orchestratore, reverse proxy, NAS service, VPN, ecc.).

---

7. Collegamento con gli appunti precedenti

Le sezioni precedenti completano e approfondiscono quanto già discusso in termini di:

- Obiettivi del cluster (bot, sito, NAS, lab).
- Riflessioni su architettura logica (master/worker, orchestratore, storage, rete).
- Strategie per accesso remoto con IP "statico" effettivo (VPS + VPN, tunneling).
- Sicurezza e autenticazione (SSH con chiavi, VPN, reverse proxy, pannelli protetti).
- Modello di trasparenza:
  - un solo punto di accesso (SSH, UI web);
  - nodi orchestrati sotto il cofano;
  - un nodo desktop o una VM come stazione di amministrazione.

Questa versione pone molto più dettaglio sulle **fasi iniziali di progettazione e valutazione**, in modo che tu possa:

- ragionare sui requisiti prima di comprare hardware;
- capire meglio quali compromessi vuoi accettare;
- scegliere un'architettura coerente con le tue reali esigenze.