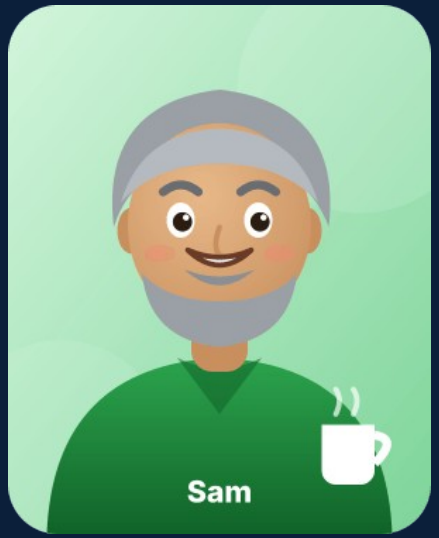
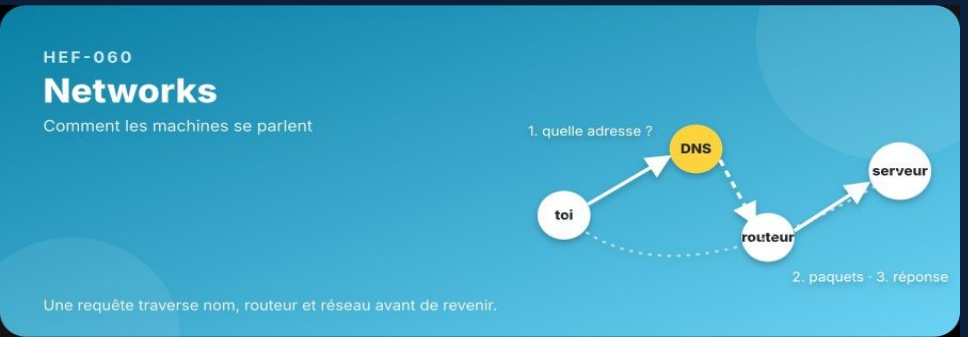
HEF-060 · MISSION 03 / 10

Adresses IP et sous-réseaux

Une adresse IP, c'est une adresse postale : un réseau (la rue) et une machine (le numéro).





Badge : Géomètre des adresses

80 min · intermédiaire

L'histoire

HashBox ouvre un second bureau. Maya veut que les ordinateurs du bureau A et ceux du bureau B soient sur des réseaux séparés. Kodi reçoit une plage d'adresses `192.168.10.0/24` et doit la découper en quatre.

> *« Pour découper, il faut comprendre ce que veut dire ce /24, dit Sam. C'est de l'arithmétique binaire que tu connais déjà. »*



Tu apprends à lire une adresse IP, un masque, et à découper un réseau en sous-réseaux.

Ta mission

À la fin, je serai capable de…

1. expliquer la structure d'une adresse IPv4 (réseau + hôte)
2. lire la notation CIDR (/24, /26…)
3. calculer adresse de réseau, de diffusion et nombre d'hôtes
4. découper un réseau en sous-réseaux
5. reconnaître les adresses privées, de boucle locale et IPv6

Défi d'échauffement

Une adresse postale : « 12 rue des Lilas, 75011 Paris ». Où se trouve la partie qui désigne **le quartier** (identique pour tous les voisins) ? Et celle qui désigne **le logement précis** ? Si une ville devait numéroter toutes les maisons avec seulement 8 bits (256 numéros), combien de rues faudrait-il pour 10 000 maisons ?

Le concept

L'adresse IPv4

Une adresse **IPv4** fait **32 bits**, écrite en 4 octets décimaux séparés par des points : `192.168.10.37`.

```
192       .168      .10       .37
11000000  10101000  00001010  00100101
```

Une adresse est composée de deux parties :

- la partie **réseau** (les premiers bits) : identique pour toutes les machines du même réseau ;
- la partie **hôte** (les derniers bits) : identifie la machine dans ce réseau.

Le masque et la notation CIDR

Le concept

Le masque et la notation CIDR (suite)

Le **masque** indique combien de bits forment la partie réseau. En **notation CIDR**, on écrit `/n` : n bits de réseau.

CIDR	Masque	Bits hôte	Adresses	Hôtes utilisables
/24	255.255.255.0	8	256	254
/25	255.255.255.128	7	128	126
/26	255.255.255.192	6	64	62
/27	255.255.255.224	5	32	30
/30	255.255.255.252	2	4	2

Hôtes utilisables = 2^(bits hôte) − 2, car on réserve :

- la première adresse : l'**adresse de réseau** (tous les bits hôte à 0) ;

Le concept

Le masque et la notation CIDR (suite)

- la dernière : l'**adresse de diffusion** (tous les bits hôte à 1).

Même réseau ou pas ?

Deux machines sont sur le même réseau si leurs adresses ont les **mêmes bits de réseau**. Sinon, elles passent par un routeur (la **passerelle**).

Adresses spéciales

Plage	Usage
`10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16`	**privées** (non routées sur internet)
`127.0.0.0/8`	boucle locale (`localhost`)
`169.254.0.0/16`	auto-adresse (pas de DHCP)

Le concept

Adresses spéciales (suite)

Plage	Usage
100.64.0.0/10	espace partagé des opérateurs (CGNAT)

IPv6

Les adresses IPv4 (4,3 milliards) ne suffisent plus. **IPv6** utilise **128 bits**, écrits en hexadécimal : `2001:0db8:85a3::8a2e:0370:7334`. Chaque réseau a généralement un préfixe de /64. Le nombre d'adresses est astronomique (2^{128}).

Le module `ipaddress`

```
import ipaddress

reseau = ipaddress.ip_network("192.168.10.0/24")
print(reseau.netmask, reseau.broadcast_address, reseau.num_addresses)
```

Le concept

Le module `ipaddress` (suite)

```
print(ipaddress.ip_address("192.168.10.37") in reseau)       # True
print(ipaddress.ip_address("10.1.2.3").is_private)           # True

for sous in reseau.subnets(new_prefix=26):
    print(sous, "→ hôtes :", sous.num_addresses - 2)
```

Exemple pas à pas

Problème : découper `192.168.10.0/24` en quatre sous-réseaux égaux pour les services Dev, Compta, Support, Invités.

Raisonnement : 4 sous-réseaux → 2 bits supplémentaires pour les distinguer → /24 + 2 = **/26** (64 adresses, 62 hôtes utilisables chacun).

```
import ipaddress

reseau = ipaddress.ip_network("192.168.10.0/24")
services = ["Dev", "Compta", "Support", "Invités"]

for nom, sous in zip(services, reseau.subnets(new_prefix=26)):
    premiere = next(sous.hosts())
    derniere = list(sous.hosts())[-1]
    print(f"{nom:8s} {str(sous):18s} hôtes {premiere} → {derniere}   diffusion {so
   us.broadcast_address}")
```

Exemple pas à pas

Exemple pas à pas (suite)

Résultat :

Service	Réseau	Hôtes	Diffusion
Dev	192.168.10.0/26	.1 à .62	.63
Compta	192.168.10.64/26	.65 à .126	.127
Support	192.168.10.128/26	.129 à .190	.191
Invités	192.168.10.192/26	.193 à .254	.255

Vérifier qu'une machine est dans le bon réseau :

```
import ipaddress

reseau_dev = ipaddress.ip_network("192.168.10.0/26")
for ip in ("192.168.10.40", "192.168.10.70"):
```

Exemple pas à pas (suite)

```
    print(ip, ipaddress.ip_address(ip) in reseau_dev)     # True puis False
```

À la main, avec le masque /26 (255.255.255.192) : le dernier octet de `192.168.10.70` en binaire est `01000110`. Les 2 premiers bits (`01`) donnent le sous-réseau n° 1 (de 64 à 127).

À toi, avec aide

Pour l'adresse `172.16.5.130/25` calcule, à la main puis avec Python :

1. l'adresse de réseau ;
2. l'adresse de diffusion ;
3. la première et la dernière adresse utilisable ;
4. le nombre d'hôtes.

Indices progressifs

Indice 1

/25 : le dernier octet est coupé en deux moitiés de 128.

Indice 2

130 est dans la deuxième moitié (128 à 255).

Indice 3

Hôtes = 128 − 2 = 126.

À toi, avec aide · réponse

Réponse

```
import ipaddress

i = ipaddress.ip_interface("172.16.5.130/25")
r = i.network
hotes = list(r.hosts())
print(r.network_address, r.broadcast_address, hotes[0], hotes[-1], len(hotes))
# 172.16.5.128 172.16.5.255 172.16.5.129 172.16.5.254 126
```

Réseau : 172.16.5.128 · diffusion : 172.16.5.255 · hôtes de .129 à .254 · 126 hôtes.

À toi, en solo

Plan d'adressage (VLSM). Tu dois répartir `10.20.0.0/24` entre :

- Serveurs : 100 machines ;
- Postes : 50 machines ;
- Wi-Fi invités : 25 machines ;
- Liaison routeur-routeur : 2 machines.

Les sous-réseaux n'ont pas la même taille (*Variable Length Subnet Masking*). Écris un programme `allouer(reseau, besoins)` qui attribue à chaque besoin le plus petit préfixe suffisant, en commençant par le plus gros besoin, sans chevauchement. Affiche le plan.

À toi, en solo · réponse

Pistes de correction

```
import ipaddress, math

def allouer(reseau, besoins):
    reseau = ipaddress.ip_network(reseau)
    courant = int(reseau.network_address)
    plan = {}
    for nom, nb in sorted(besoins.items(), key=lambda x: -x[1]):
        bits_hote = max(2, math.ceil(math.log2(nb + 2)))
        prefixe = 32 - bits_hote
        sous = ipaddress.ip_network((courant, prefixe))
        if not sous.subnet_of(reseau):
            raise ValueError(f"plus de place pour {nom}")
        plan[nom] = sous
        courant = int(sous.broadcast_address) + 1
    return plan

plan = allouer("10.20.0.0/24", {"Serveurs": 100, "Postes": 50, "Invités": 25, "Lia
   ison": 2})
for nom, sous in plan.items():
```

À toi, en solo · réponse

Pistes de correction (suite)

```
    print(f"{nom:9s} {sous}  ({sous.num_addresses - 2} hôtes)")
```

Attendu : Serveurs /25 (126 hôtes), Postes /26 (62), Invités /27 (30), Liaison /30 (2). Allouer du plus gros au plus petit évite la fragmentation et garantit l'alignement des blocs.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Oublier les 2 adresses réservées	254 hôtes pour un /24, pas 256	Hôtes = 2^bits − 2
Mal aligner un sous-réseau	Un /26 doit commencer à un multiple de 64	Allouer du plus grand au plus petit
Chevauchement	Deux réseaux se superposent	Vérifier avec `overlaps()`
Router des adresses privées sur internet	Elles ne sont pas joignables	NAT (mission 9)

Mini-boss

Outil d'adressage. Écris `analyser(cidr)` qui renvoie un dictionnaire complet : adresse de réseau, masque, joker (wildcard), diffusion, première/dernière adresse, nombre d'hôtes, type (privée, publique, boucle locale, lien local, multicast), binaire du masque.
Puis écris `meme_reseau(ip1, ip2, prefixe)` **sans** utiliser `ipaddress.ip_network` (uniquement des opérations binaires sur les entiers), et vérifie-la contre la version bibliothèque sur 1 000 couples aléatoires.

Mini-boss · réponse

Pistes de correction

```
import ipaddress, random

def analyser(cidr):
    i = ipaddress.ip_interface(cidr)
    r = i.network
    hotes = list(r.hosts()) if r.num_addresses > 2 else []
    a = i.ip
    type_ = ("boucle locale" if a.is_loopback else "lien local" if a.is_link_local
             else "multicast" if a.is_multicast else "privée" if a.is_private else
    "publique")
    return {
        "reseau": str(r.network_address), "masque": str(r.netmask),
        "joker": str(r.hostmask), "diffusion": str(r.broadcast_address),
        "premiere": str(hotes[0]) if hotes else None, "derniere": str(hotes[-1]) i
   f hotes else None,
        "hotes": max(r.num_addresses - 2, 0), "type": type_,
        "masque_binaire": format(int(r.netmask), "032b"),
    }
```

Mini-boss · réponse

Pistes de correction (suite)

```
def meme_reseau(ip1, ip2, prefixe):
    masque = (0xFFFFFFFF << (32 - prefixe)) & 0xFFFFFFFF
    return (int(ipaddress.ip_address(ip1)) & masque) == (int(ipaddress.ip_address(
   ip2)) & masque)

print(analyser("192.168.10.37/26"))
rng = random.Random(7)
for _ in range(1000):
    a = ipaddress.ip_address(rng.getrandbits(32))
    b = ipaddress.ip_address(rng.getrandbits(32) if rng.random() < 0.5 else int(a)
    ^ rng.getrandbits(6))
    p = rng.randint(8, 30)
    attendu = b in ipaddress.ip_network((int(a) & ((0xFFFFFFFF << (32 - p)) & 0xFF
   FFFFFF), p))
    assert meme_reseau(str(a), str(b), p) == attendu
print("ok")
```

Quiz éclair

Quiz éclair

1. Un réseau /24 contient combien d'hôtes utilisables ?

A. 256
B. 254
C. 255

2. Le masque 255.255.255.192 correspond à :

A. /24
B. /28
C. /26

3. Une adresse IPv6 est composée de :

A. 128 bits

Quiz éclair

Quiz éclair (suite)

B. 32 bits
C. 64 bits

4. 🔁 *Révision mission 02* : Une adresse MAC est composée de :

A. 32 bits
B. 48 bits
C. 128 bits

5. 🔁 *Révision mission 02* : Un switch apprend où se trouvent les machines en observant :

A. l'adresse IP de destination
B. les ports des programmes
C. l'adresse MAC source des trames

Quiz éclair · réponse

Réponses

1. **B**.
2. **C**.
3. **A**.
4. **B**.
5. **C**.

Je réfléchis

1. Dans ta vie, où retrouves-tu un système « adresse = zone + numéro » (code postal, téléphone) ?
2. Pourquoi découper un réseau en sous-réseaux améliore-t-il la sécurité et la gestion ?

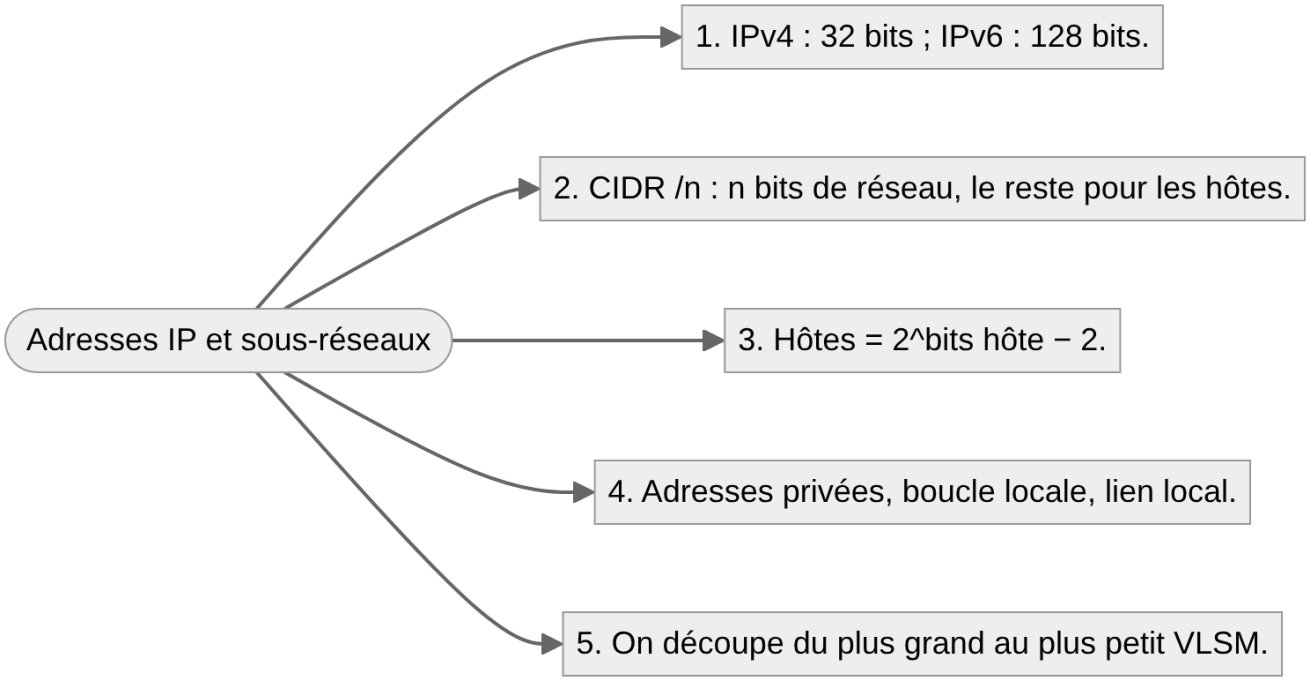
À retenir

À retenir

1. IPv4 : 32 bits ; IPv6 : 128 bits.
2. CIDR /n : n bits de réseau, le reste pour les hôtes.
3. Hôtes = 2^bits hôte − 2.
4. Adresses privées, boucle locale, lien local.
5. On découpe du plus grand au plus petit (VLSM).

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Géomètre des adresses

Prochaine mission : Le routage

