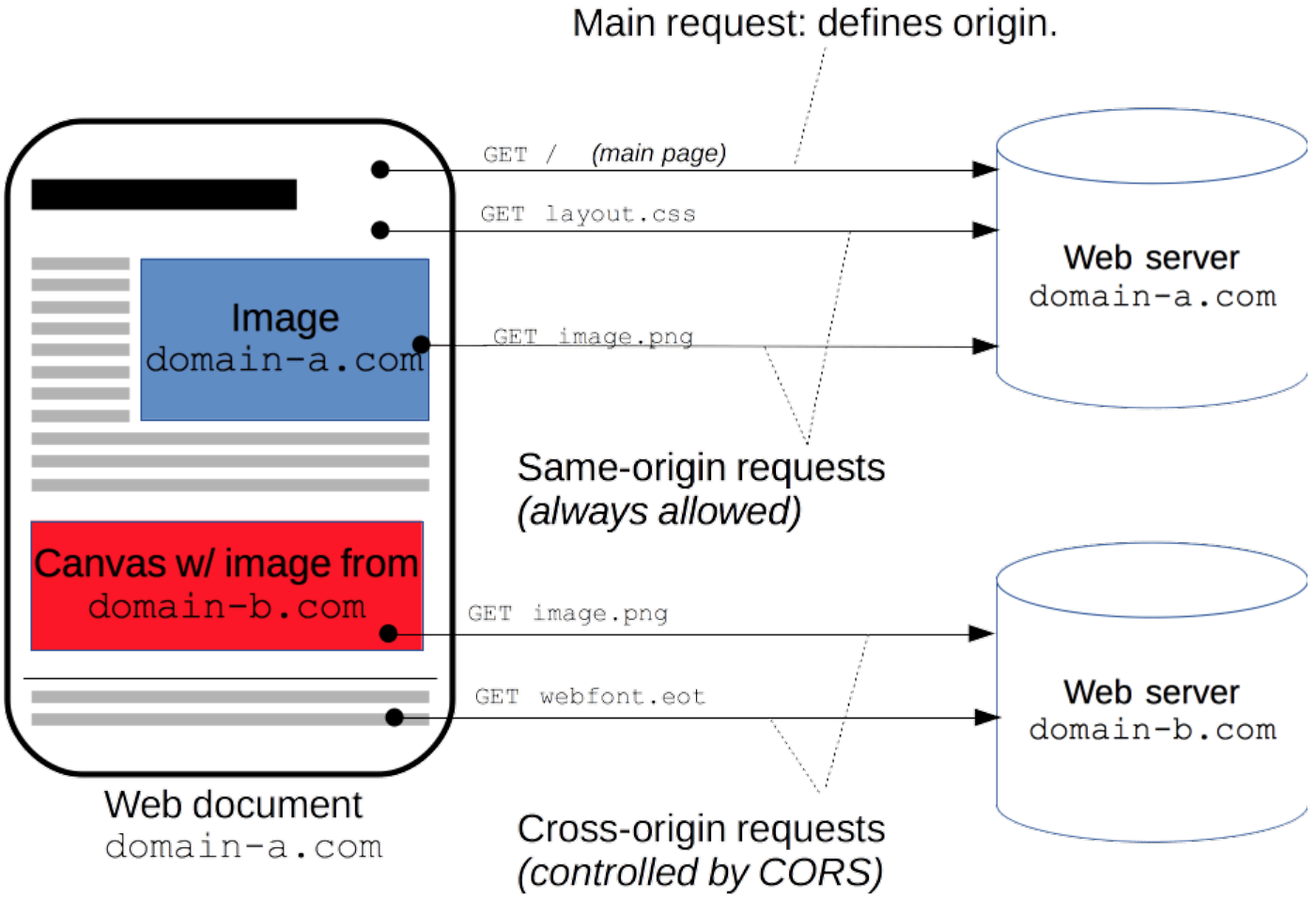
Gestion des Headers Cors : X-origin requests

- **Problème** : la contrainte "same origin" imposée par les navigateurs :

  Une application web peut uniquement émettre des requêtes vers la même origine que celle à partir de laquelle l'application à été chargée.

- Cette contrainte de sécurité est **appliquée par le navigateur** qui interdit l'utilisation de données d'origine tierce et concerne
  - les requêtes fetch et xhr
  - les polices @font-face
  - les frames dessinées sur un canevas



https://developer.mozilla.org/fr/docs/Web/HTTP/CORS

CORS

- **CORS** (*Cross Origin Resource Sharing*) est un mécanisme pour lever cette contrainte et autoriser des requêtes X-origin de façon contrôlée par le serveur
- Le serveur indique l'origine et les caractéristiques des requêtes autorisées – *il peut ainsi interdire par exemple des requêtes émises par une origine douteuse*
- Le mécanisme est basé sur l'ajout de headers dédiés dans les requêtes/réponses entre le client et le serveur

Gestion des headers CORS

- De façon automatique par le navigateur pour les headers de requêtes
- Doit être programmé du côté serveur pour les headers de réponses

- Doit impérativement être traité lors de l'implantation d'une API pour qu'elle soit utilisable
- Attention : il s'agit d'une contrainte imposée par les navigateurs uniquement
  - pas détectable avec des outils type cURL, postman...

CORS : 2 cas

Requêtes "Simples"

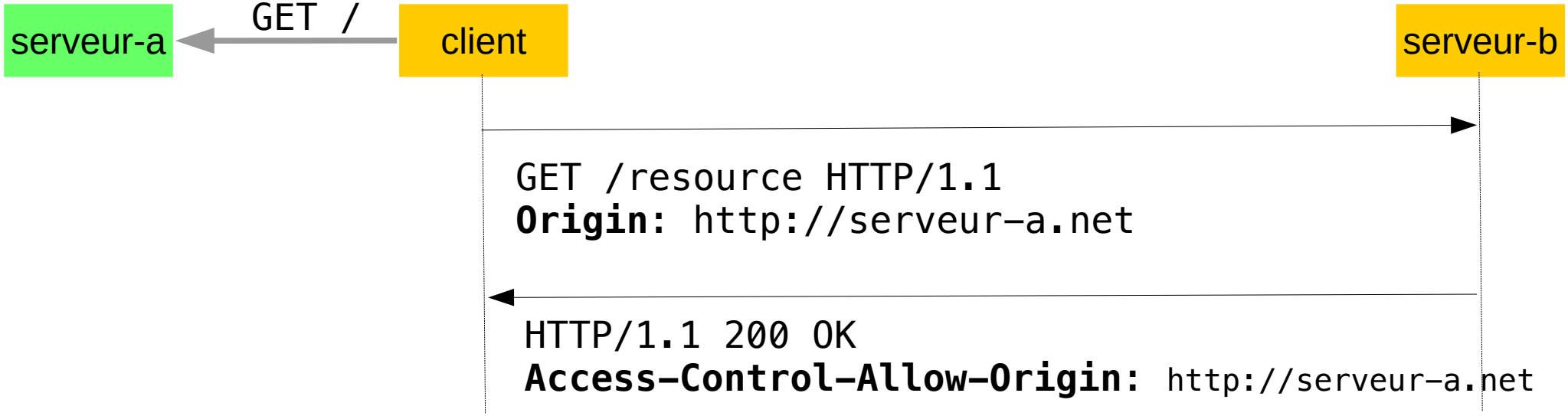
- GET, POST, HEAD
- **ET**
- headers positionnés par le navigateur +
- Accept, Accept-Language, Content-Language
- Content-Type : x-www-form-url-encoded / form-data / text-plain

Requêtes "Contrôlées"

- Autres méthodes
- **OU**
- Autres headers, dont
- Authorization
- headers applicatifs X-xxx
- Content-Type avec une autre valeur : application/json

Cas des requêtes "simples"

- Le client ajoute un header indiquant l'origine de la requête
- Le serveur répond en plaçant un header indiquant les origines autorisées



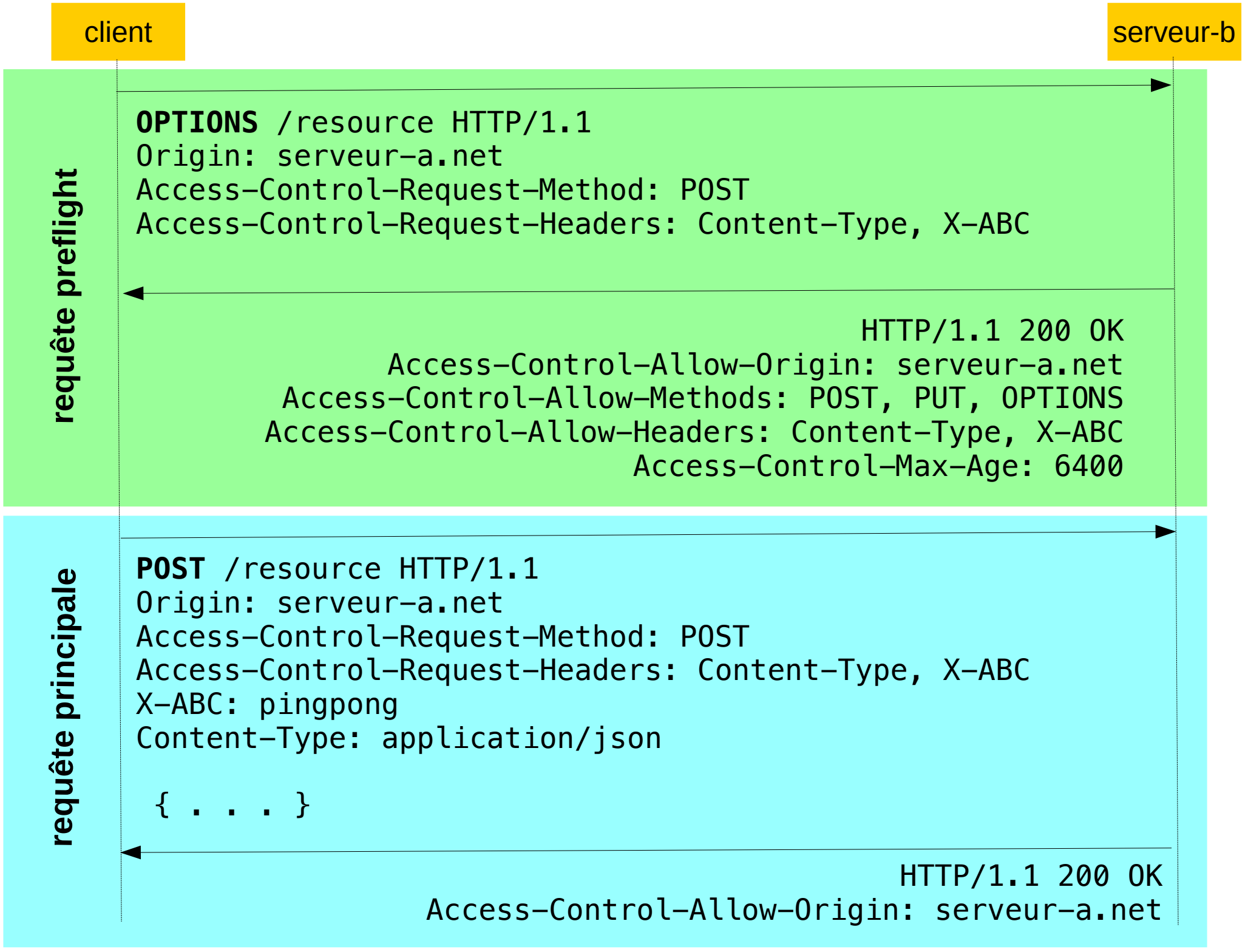
- Un serveur exposant des ressources publiques peut répondre '*'

En PHP : dans un middleware

```php
public function corsHeaders(
        Request $rq,
        RequestHandllerInterface $next ): Response {

// Si on veut être très strict : exiger la présence du header Origin dans la
// requête
    if (! $rq->hasHeader('Origin'))
        New HttpUnauthorizedException ($rq, "missing Origin Header (cors)");

    $response = $next->handle($rq);

    $response =  $response
     ->withHeader('Access-Control-Allow-Origin', $rq->getHeader('Origin'));

    // ou bien, pour un serveur public :
        ->withHeader('Access-Control-Allow-Origin', '*');

   // pour une liste d'origines fixe et identifiée :
        ->withHeader('Access-Control-Allow-Origin', 'http://myapp.net')
        ->withHeader('Access-Control-Allow-Origin', 'http://myapp.com');;

    return $response;
}
```

Cas des requêtes contrôlées

- Une requête contrôlée est une requête réalisée avec 2 requêtes HTTP :
  - une pré-requête de contrôle (***preflight***) indiquant l'origine, la méthode prévue, les headers utilisés
  - le serveur répond en autorisant ou pas la demande,
  - la requête effective, envoyée si la réponse du serveur autorise la demande,
  - le serveur répond en autorisant l'origine


client
serveur-b
requête preflight
OPTIONS /resource HTTP/1.1
Origin: serveur-a.net
Access-Control-Request-Method: POST
Access-Control-Request-Headers: Content-Type, X-ABC
HTTP/1.1 200 OK
Access-Control-Allow-Origin: serveur-a.net
Access-Control-Allow-Methods: POST, PUT, OPTIONS
Access-Control-Allow-Headers: Content-Type, X-ABC
Access-Control-Max-Age: 6400
requête principale
POST /resource HTTP/1.1
Origin: serveur-a.net
Access-Control-Request-Method: POST
Access-Control-Request-Headers: Content-Type, X-ABC
X-ABC: pingpong
Content-Type: application/json
{ . . . }
HTTP/1.1 200 OK
Access-Control-Allow-Origin: serveur-a.net

Implantation de l’API

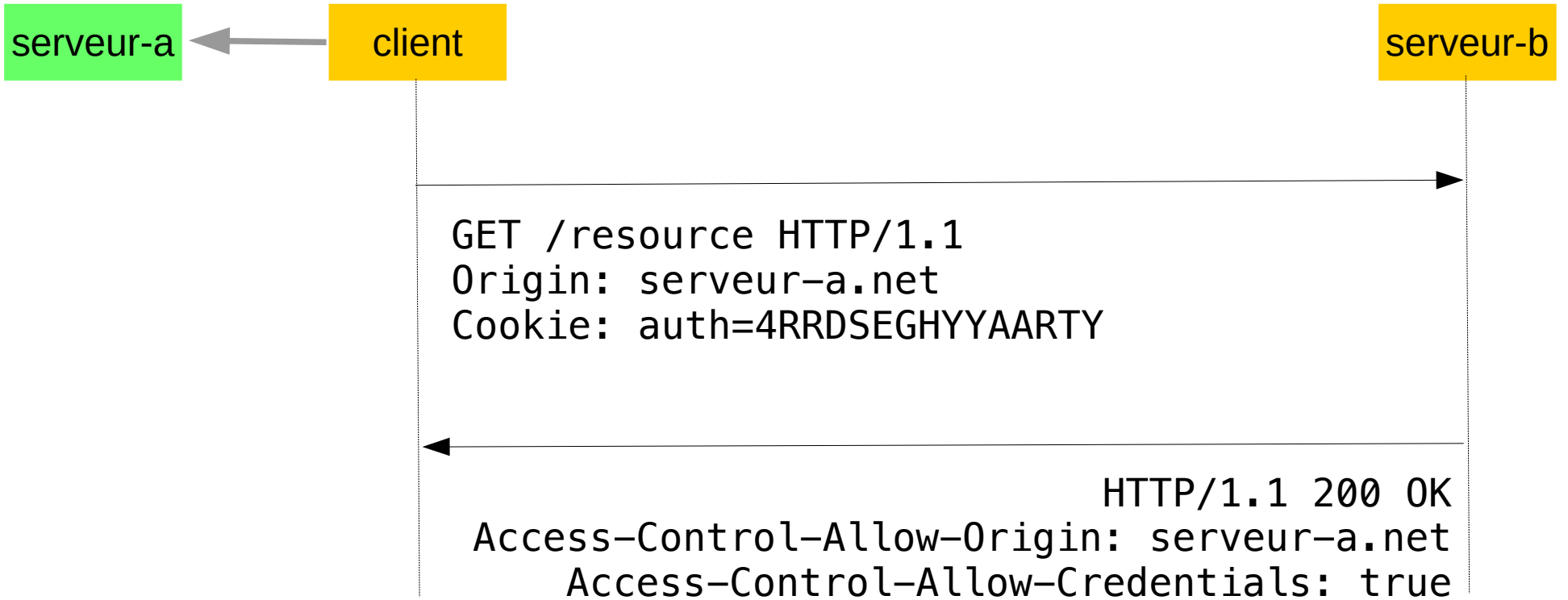
- L’API DOIT implanter la méthode OPTIONS sur toutes les routes concernées par un preflight

Requêtes avec credentials

- Par défaut, le navigateur ne positionne pas les credentials dans une requête xhr/fetch X-domain
  - credentials = cookies ou Header Authorization
- Pour dépasser cette limitation il faut :

1) du côté de l'application cliente, créer la requête avec le flag "`withCredentials`"
2) du côté serveur, répondre avec le header

```
Access-Control-Allow-Credentials: true
```


serveur-a
client
serveur-b
GET /resource HTTP/1.1
Origin: serveur-a.net
Cookie: auth=4RRDSEGHYYAARTY
HTTP/1.1 200 OK
Access-Control-Allow-Origin: serveur-a.net
Access-Control-Allow-Credentials: true

Headers de requêtes

- `Origin` : obligatoire dans toute requête CORS, indique le domaine d'origine du document (page, application) émettant la demande,
- `Access-Control-Request-Method` : (requête preflight), indique la méthode prévue pour la requête principale
- `Access-Control-Request-Headers` : (requête preflight), indique les headers prévus dans la requête principale

Headers de réponses

- `Access-Control-Allow-Origin:` * | domaine
- `Access-Control-Allow-Methods:` Liste
- `Access-Control-Allow-Headers:` Liste
- `Access-Control-Allow-Credentials:` true
- `Access-Control-Expose-Headers:` Liste de headers placés par le serveur que le client peut lire
- `Access-Control-Max-Age:` Durée de validité de la requête preflight

Implantation minimum

- 1 middleware contrôlant le header Origin sur toutes les requêtes
- 1 middleware plaçant les headers de réponses sur toutes les réponses
  - dans de nombreux cas, on peut se contenter d'une réponse identique pour toutes les requêtes
- 1 route générique sur la méthode OPTIONS répondant à toutes les requêtes preflight

En PHP/slim

- Voir la doc, chapitre "enabling CORS"
- Il existe des middleware PSR-15 dédiés
- Compléter le middleware pour ajouter tous les headers nécessaires + traiter les requêtes OPTIONS

```php
/**
 *   ajouter le middleware CORS sur toutes les routes
*/
$app->add(new \middlewares\Cors::class ) ;
/*
* déclarer les routes option – il suffit de répondre en ajoutant les
* headers grâce qu middleware
*/
$app->options('/{routes:.+}',
       function( Request $rq,
                 Response $rs, array $args) : Response {

    return $rs;
});
```

```php
public function corsHeaders(Request $rq,
                            RequestHandlerInterface $next ): Response {

  if (! $rq->hasHeader('Origin'))
    New HttpUnauthorizedException ($rq, "missing Origin Header (cors)");

  $response = $next->handle($rq);

  $response =  $response
        ->withHeader('Access-Control-Allow-Origin', 'http://myapp.net')
        ->withHeader('Access-Control-Allow-Methods', 'POST, PUT, GET' )
        ->withHeader('Access-Control-Allow-Headers','Authorization' )
        ->withHeader('Access-Control-Max-Age', 3600);
        ->withHeader('Access-Control-Allow-Credentials', 'true');

    return $response;
}
```