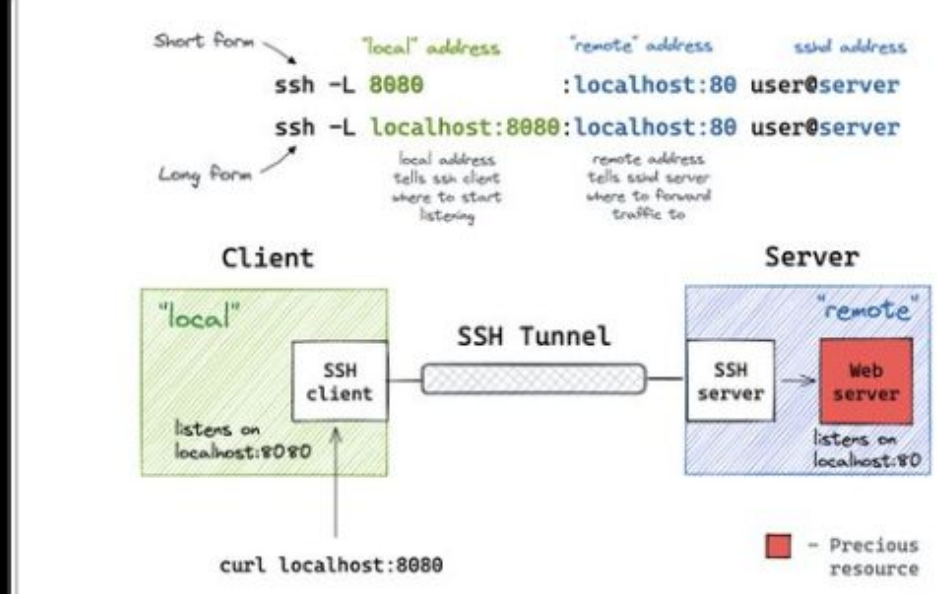
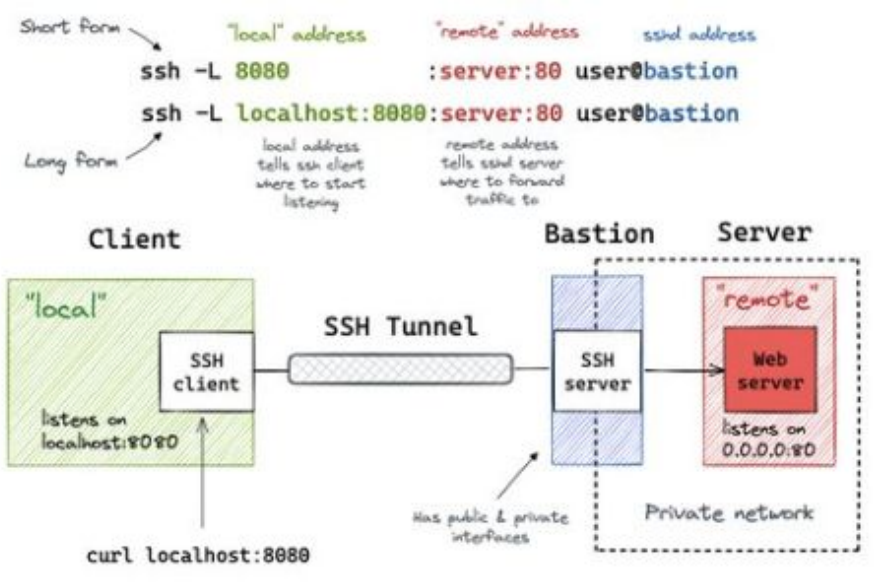
SSH Port Forwarding: Why and How 🧵

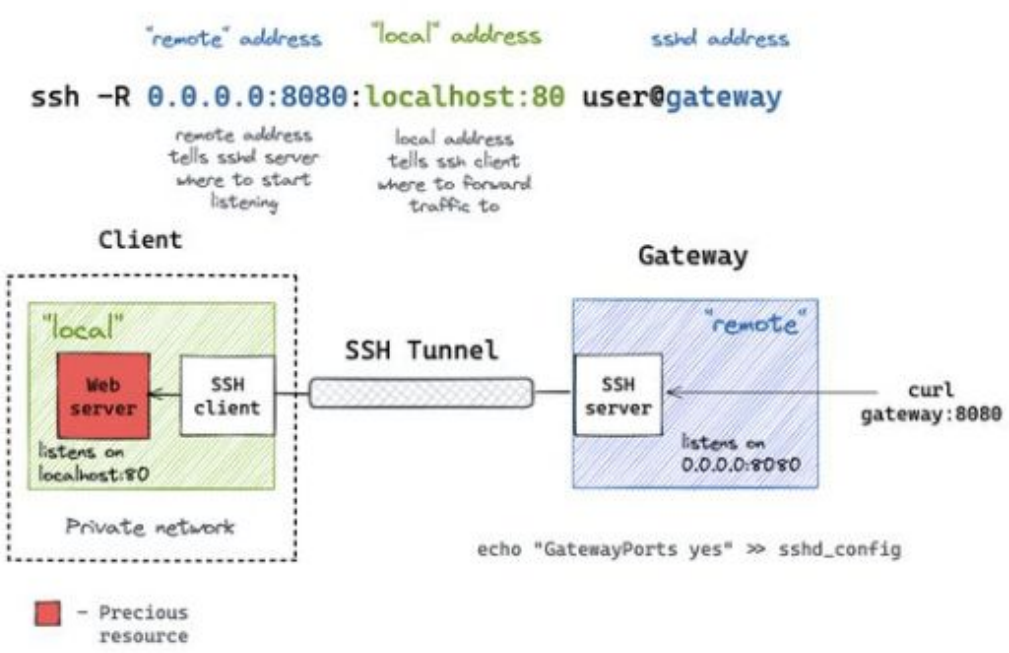
If these problems sound familiar:

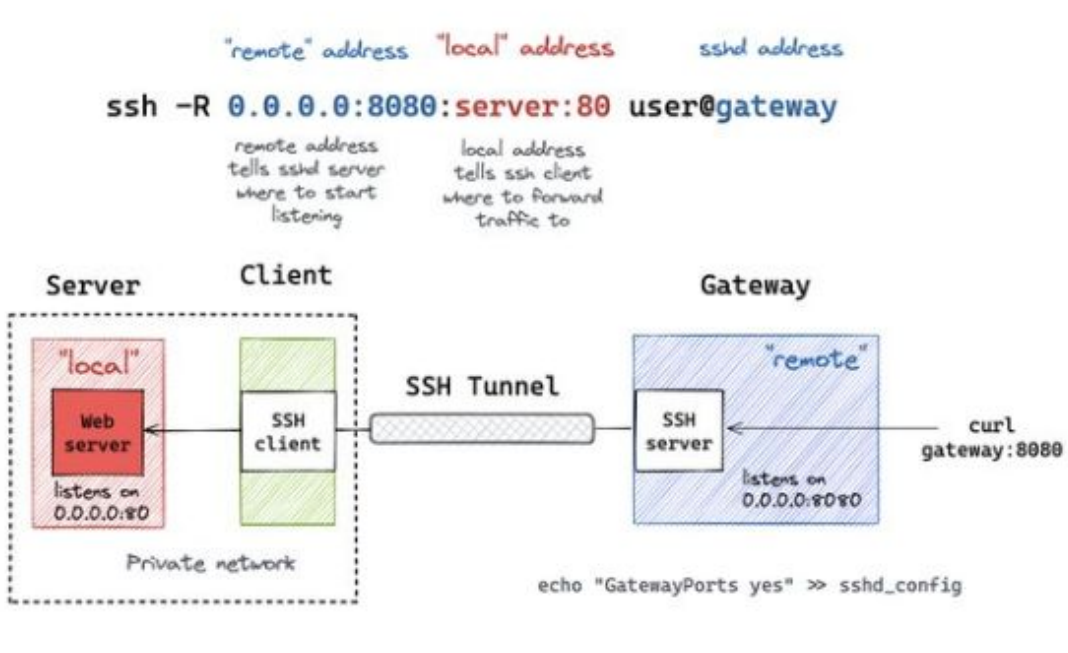
- A db server listens on a remote localhost, but you want to use a local GUI client
- A dev service runs on your laptop, but you want to expose it to the Internet

...and you don't know the solution, read on!





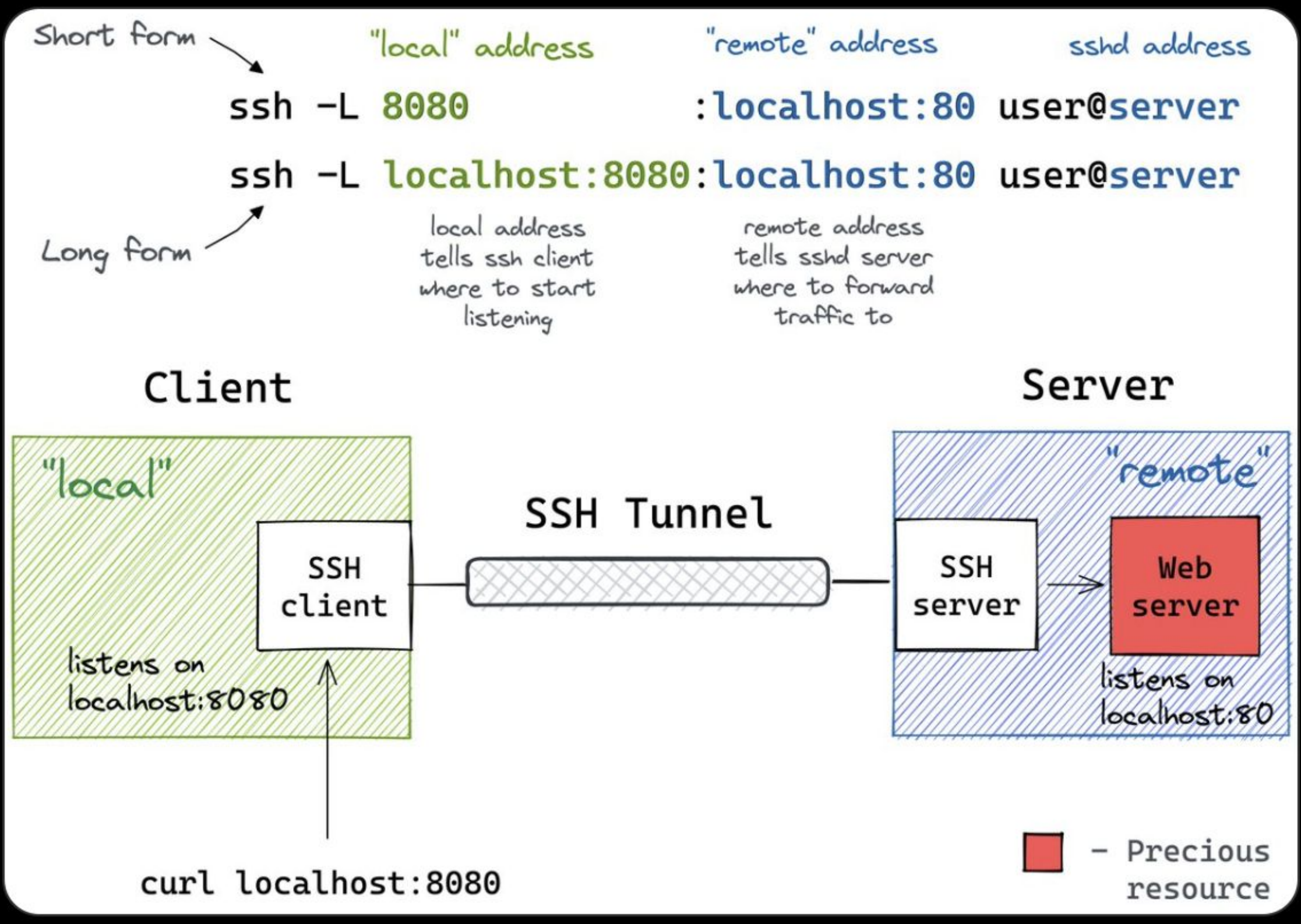




1. Simple Local Port Forwarding.

If you have SSH access to the target server, you can bind any port from this server to your local port.

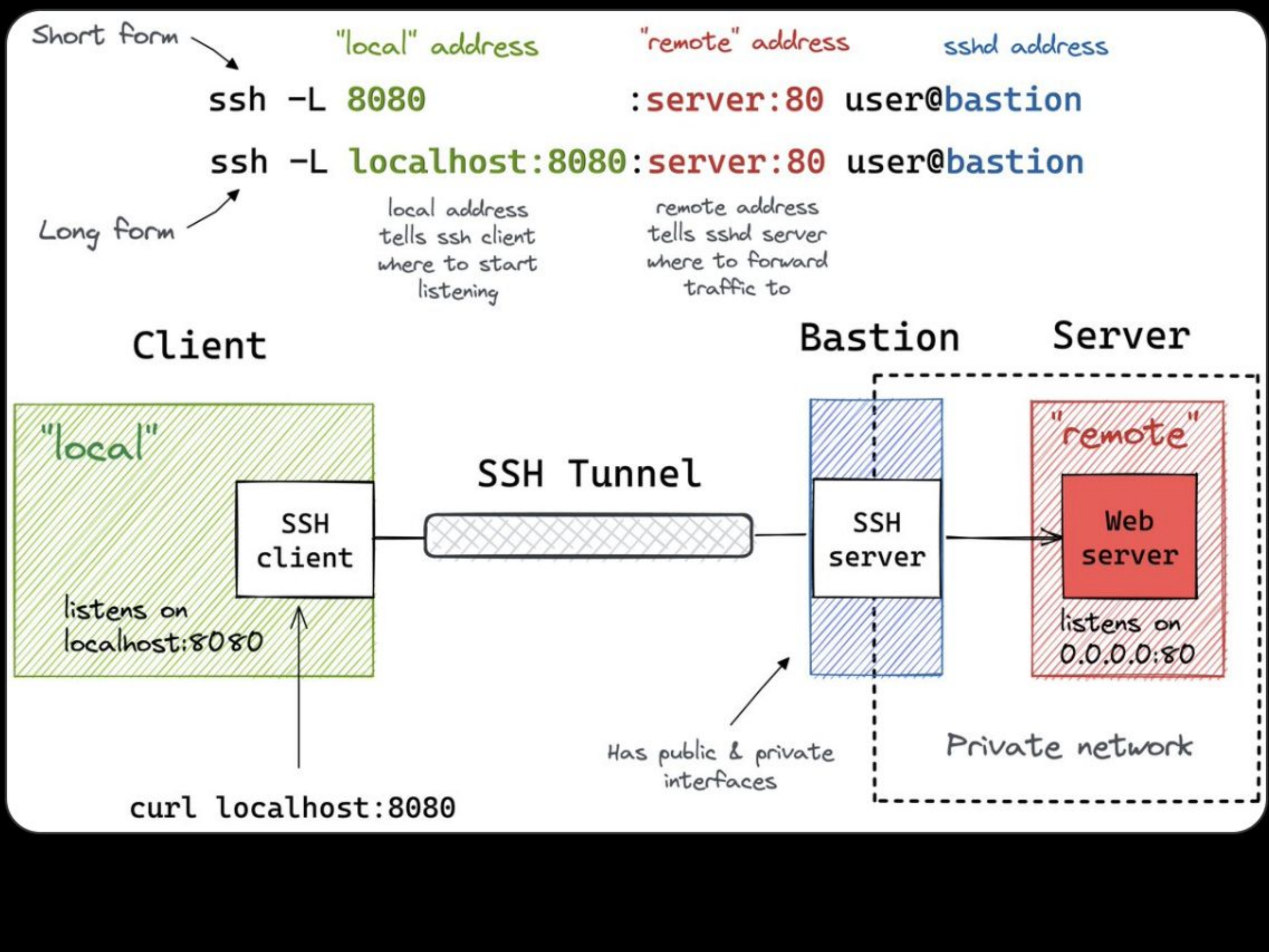
Run the SSH client with the -L flag to make it listen on a local port. Any traffic to this port will be forwarded to the SSH server.



2. Advanced Local Port Forwarding.

Sometimes, the target is not on the server you have SSH access to but on some other "upstream" host behind it.

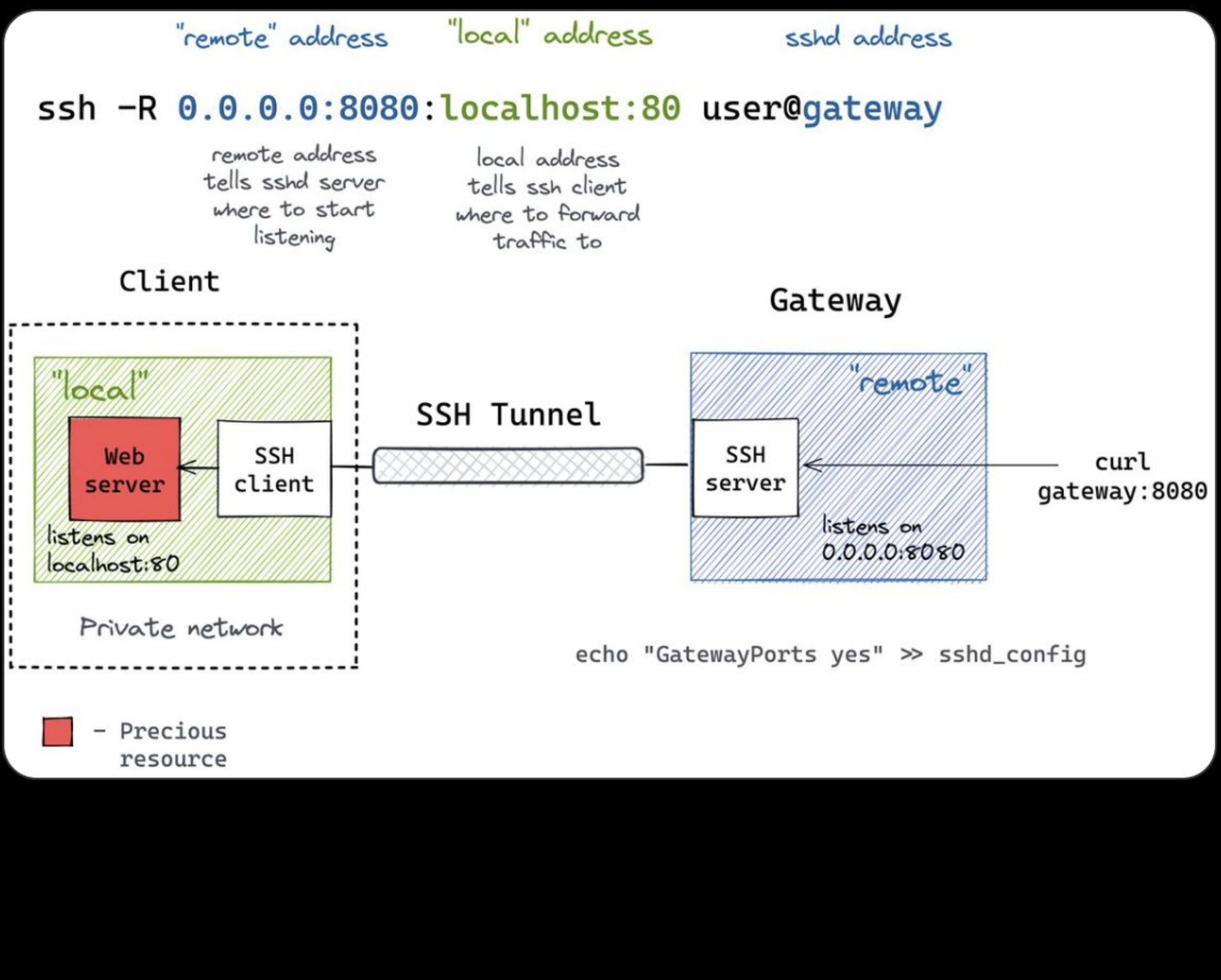
Example: An EC2 instance with public and private interfaces. You can SSH to one and access private VPC resources through another.



3. Remote Port Forwarding.

Remote port forwarding solves the inverse problem.

If you have a local resource (e.g., a dev server running on your laptop) and you have SSH access to a public-facing server, you can expose the resource to the Internet with just one "ssh -R" command.



4. Remote Port Forwarding to Home/Private Network.

In the case of remote port forwarding, the new port is opened by the SSH server (sshd). The SSH server then forwards any traffic destined for this port to your SSH client. But the SSH client itself can forward traffic further!