Hashicorp Boundary Investigation Report

1. License

Mozilla Public License 2.0

<https://www.mozilla.org/en-US/MPL/2.0/>

MPL既是得到自由软件基金会承认的自由软件许可证[3]，也是得到开放源代码促进会承认的开源软件许可证[4]。MPL允许在其授权下的源代码与其他授权的文件进行混合，包括私有许可证。但在MPL授权下的代码文件必须保持MPL授权，并且保持开源。[5]这样的条款让MPL既不像MIT和BSD那样允许派生作品完全转化为私有，也不像GPL那样要求所有的派生作品，包括新的组件在内，全部必须保持GPL。通过允许在派生项目中存在私有模块，同时保证核心文件的开源，MPL同时激励了商业及开源社区来参与帮助开发核心软件。[14]

使用MPL许可的软件并不受专利的限制，其可以自由使用，出售，并可自由的重新发布。带有专利代码的版本仍然可以使用，转让，甚至出售，但未经许可则不能修改代码。此外，MPL并不授予用户对于开发者商标的使用权[5]。

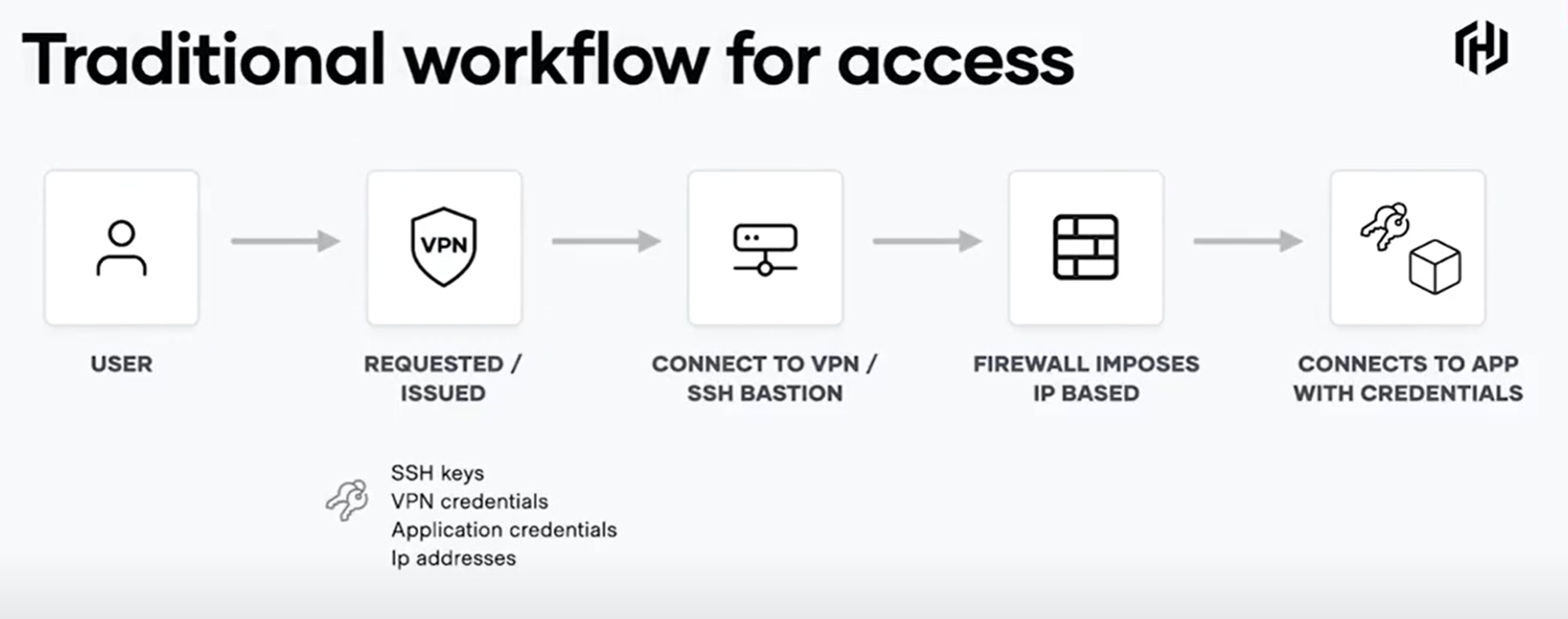
为了满足MPL的条款限制，用户必须负担一些“责任”，主要是关于散发使用MPL许可的软件。用户必须确保重新散发的软件所有源代码均以MPL许可，即使是以可执行文件的方式提供或是与其他使用专有软件许可的源代码结合也一样。但若跟以GNU通用公共许可协议、GNU宽通用公共许可证、Affero通用公共许可证许可的源代码结合则是例外。此时开发者则可选用以上三种更加严格的条款来许可[5]。

1. Introduction
   1. Concept

Boundary is a new secure access management offering from HashiCorp, that's going to enable identity-based access controls for dynamic infrastructure.

<https://www.boundaryproject.io/docs/concepts>

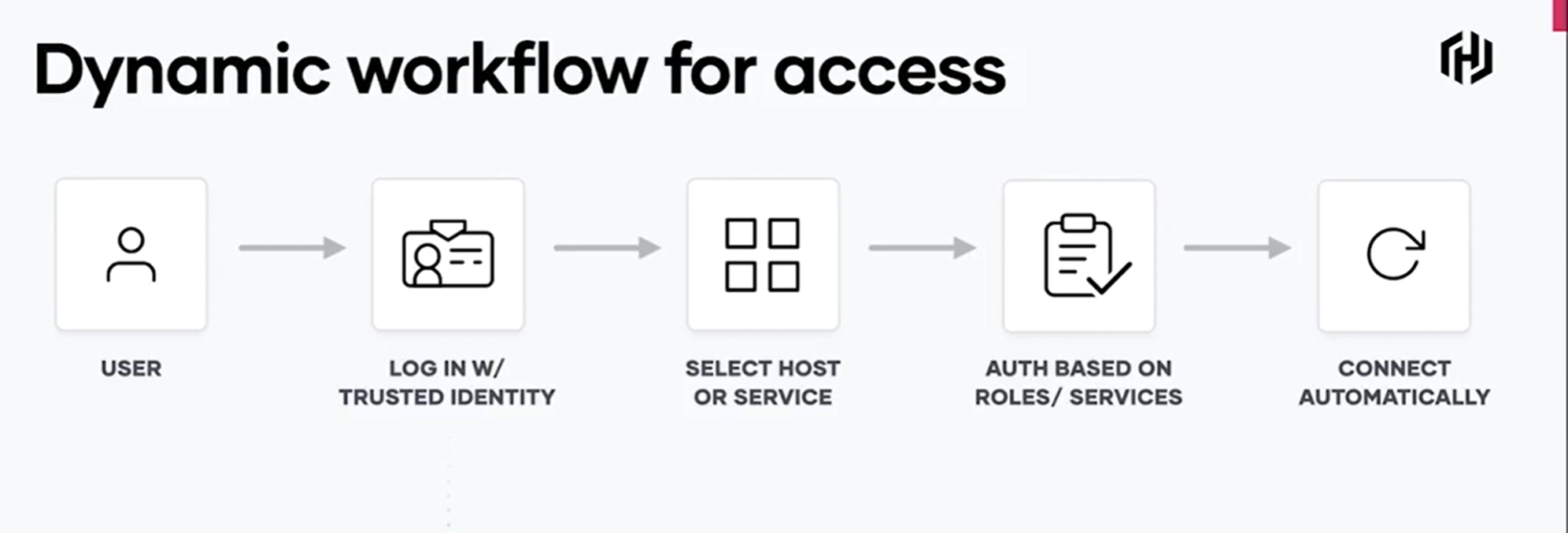
**Traditional workflow for access**



**Dynamic workflow for access**

With Boundary, access permissions are based on the user client's identity rather than their network location or their access to static credentials. Their identity is assigned to a role which gives it access to a set of hosts and services that it can connect to and perform actions on securely — even over lower zero trust networks.

When I go to connect, the connection configuration to that end target is automated. This means the process of discovering new targets as they're deployed is automated — as is the process of applying rules, policies, and credentials that govern how I can access my target. This way, the access policies assigned to that target are applied even as new host instances are provisioned or updated.



* 1. **Boundary’s Structure and Hierarchy**

<https://www.hashicorp.com/resources/secure-access-management-with-hashicorp-boundary>

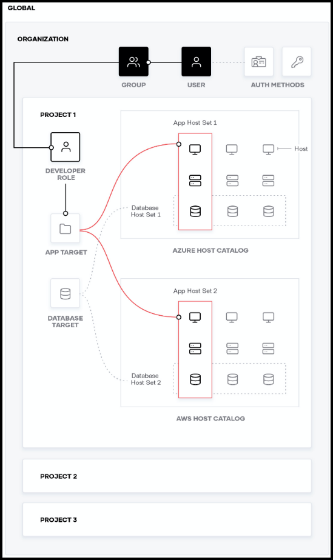
<https://medium.com/hashicorp-engineering/hashicorp-boundary-make-sure-your-human-to-machine-access-is-secure-68718674c22c>

**HashiCorp Boundary’s Structure and Hierarchy**

<https://www.boundaryproject.io/docs/concepts/domain-model>

Below is an image showing the general structure within Boundary.

At the top level is a Global.

 Diagram

Description automatically generated with low confidence

* An Organization is a child of Global.
* Within an Organization, there are auth methods, users, groups, and projects.
* Within each project, you define roles, targets, and host catalogs.
* Inside a host catalog are host sets.
* Inside of host sets are the end target hosts.

You can create host sets that contain application infrastructure such as App Host Set 1 shown below. You can also create host sets based on function, such as the database host set shown in the image below. This way, database admins can have access to all databases in the organization.

* 1. 部署环境需求

**Dev**

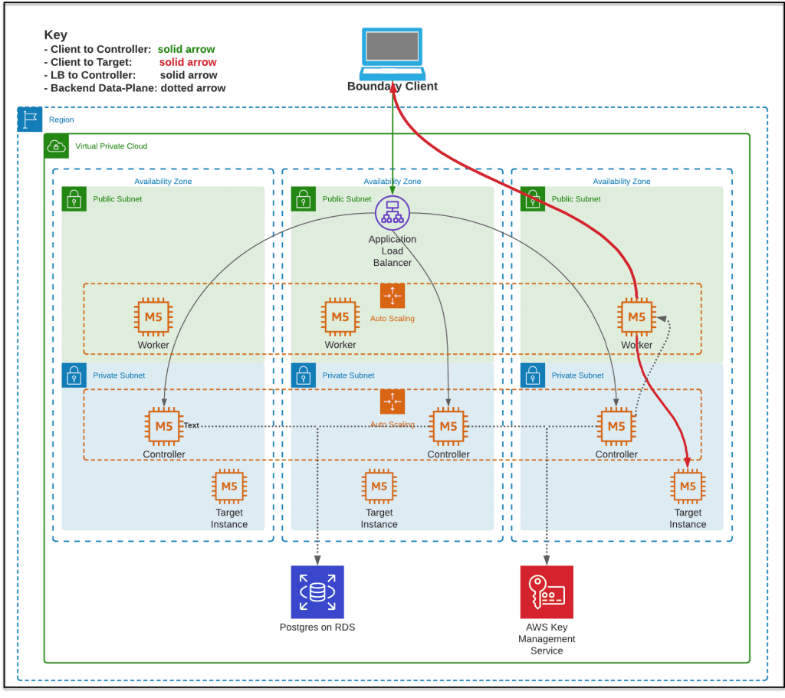
<https://learn.hashicorp.com/tutorials/boundary/getting-started-install>

<https://learn.hashicorp.com/tutorials/boundary/getting-started-dev?in=boundary/getting-started>

**Production Deployment**

(已经和得到项目开发人员的确认，目前该项目需要安装boundary在每一个client上。<https://discuss.hashicorp.com/t/do-we-need-to-install-boundary-to-every-client/18218>)

Below is a production deployment architectural diagram. Boundary has controllers and workers. In dev mode, they are on the same machine, but in production they’re separate. In short, controllers expose the management plane via an API, whereas workers are the brokers or proxies of the secure sessions. <https://www.boundaryproject.io/docs/installing/high-availability>



* 1. 开源社区成熟度

1. 与已有项目融合开发
   1. Go language
   2. Commercial support