Putting all resources related to a gateway, microservices, databases, and other components under a single resource group is a common practice in Azure, and it has several advantages and some potential downsides. Here's a detailed look at both:

Advantages

1. **Simplified Management** :

- **Unified Management** : All related resources can be managed together, making it easier to apply policies, monitor resources, and perform operations such as backup and disaster recovery.
- **Single Pane of Glass** : Viewing and managing all resources from a single resource group can simplify operations and provide a clear overview of the application's components.

2. **Resource Group Level Operations** :

- **Bulk Actions** : You can perform bulk actions such as starting, stopping, or deleting all resources in the resource group simultaneously.
- **Access Control** : Managing access control is simplified as you can assign roles and permissions at the resource group level rather than on individual resources.

3. **Deployment and Automation** :

- **Deployment Automation** : Infrastructure as Code (IaC) tools like ARM templates, Terraform, and Ansible can deploy all resources in a single resource group, making automated deployments easier.
- **Consistent Configuration** : Ensures that all related resources are consistently configured and deployed together.

4. **Billing and Cost Management** :

- **Cost Tracking** : It becomes easier to track and manage costs for all resources related to a specific application or service since all related costs are aggregated under a single resource group.

Potential Downsides

1. **Resource Limits** :

- **Quota Limits** : Resource groups have certain limits on the number of resources (e.g., VMs, IP addresses) they can contain. If your deployment is very large, you may run into these limits.
- **Management Overhead** : Managing a very large number of resources within a single group can become cumbersome.

2. **Security and Access Control** :

- **Granular Permissions** : If different teams need access to different sets of resources, managing granular permissions can become complex when all resources are in a single group.

3. **Organizational Constraints** :

- **Organizational Boundaries** : If your organization prefers to separate resources by function, environment (e.g., dev, test, prod), or department, placing all resources in a single group may not align with these organizational boundaries.

4. **Disaster Recovery** :

- **Recovery Scope** : In case of a disaster recovery scenario, it might be necessary to recover only specific parts of the application. Having everything in a single resource group could complicate such selective recovery efforts.

Recommendations

1. **Logical Grouping** :

- Group related resources together logically. For example, consider creating separate resource groups for different environments (e.g., `rg-app-prod`, `rg-app-dev`), or for major functional components (e.g., `rg-app-gateway`, `rg-app-services`).

2. **Access Control and Security** :

- Use resource groups to enforce security boundaries. Apply role-based access control (RBAC) at the resource group level to manage who can access which resources.

3. **Consider Limits** :

- Be aware of Azure's resource group limits. If your deployment is large, plan to distribute resources across multiple resource groups to avoid hitting these limits.

4. **Resource Tagging** :

- Use tags to organize and manage resources across resource groups. Tags can help in identifying resources that belong to a specific application, environment, or cost center.

Conclusion

Using a single resource group for all related resources is a viable strategy and can simplify management, especially for small to medium-sized deployments. However, consider the potential downsides and plan your resource organization according to your specific needs, organizational policies, and Azure best practices.