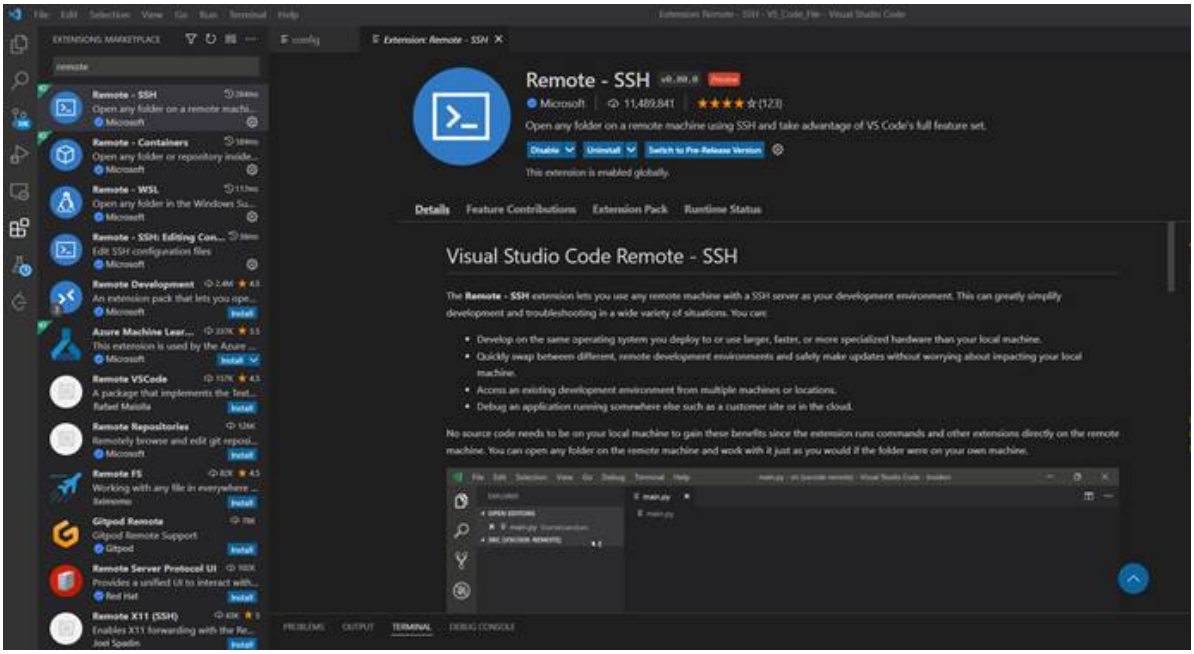
VSCode远程修改及编译过程

VSCode版本号：1.67.2

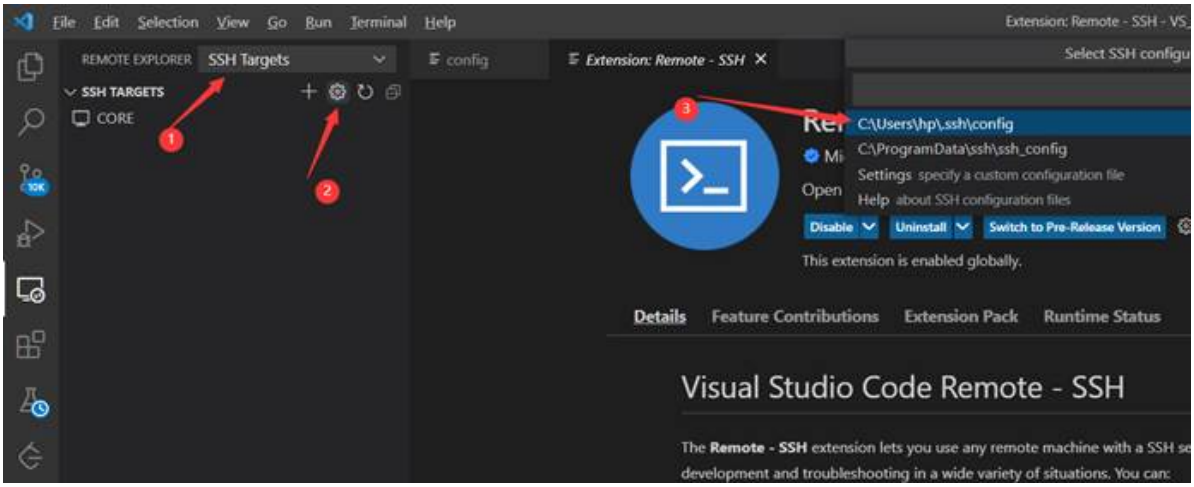
Linux内核版本：5.4.1

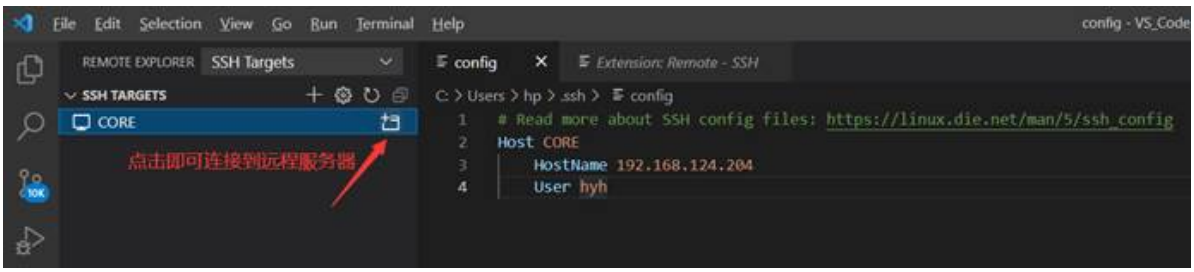
安装SSH插件

SSH插件版本号：v0.80.0

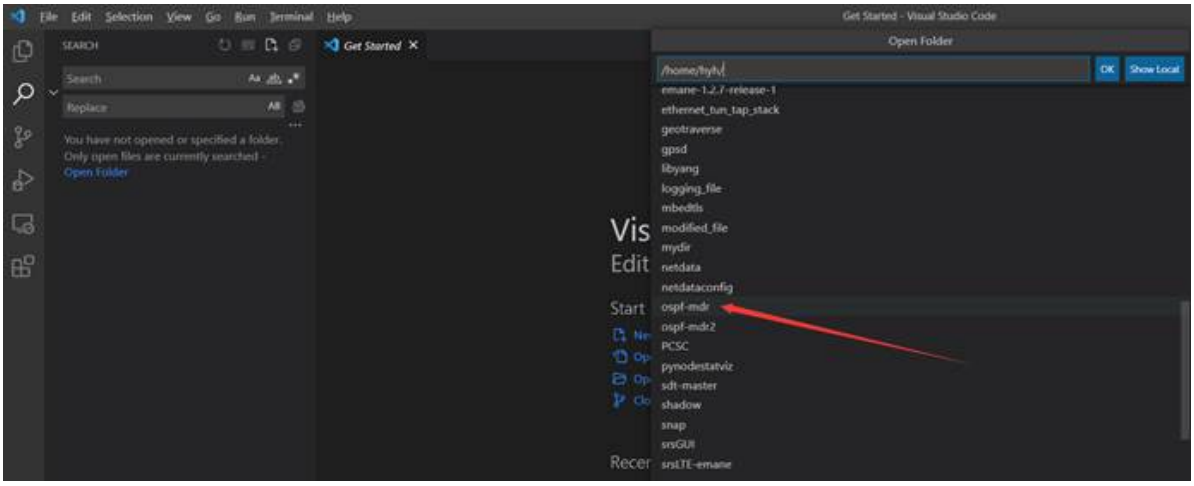


配置远程服务器信息

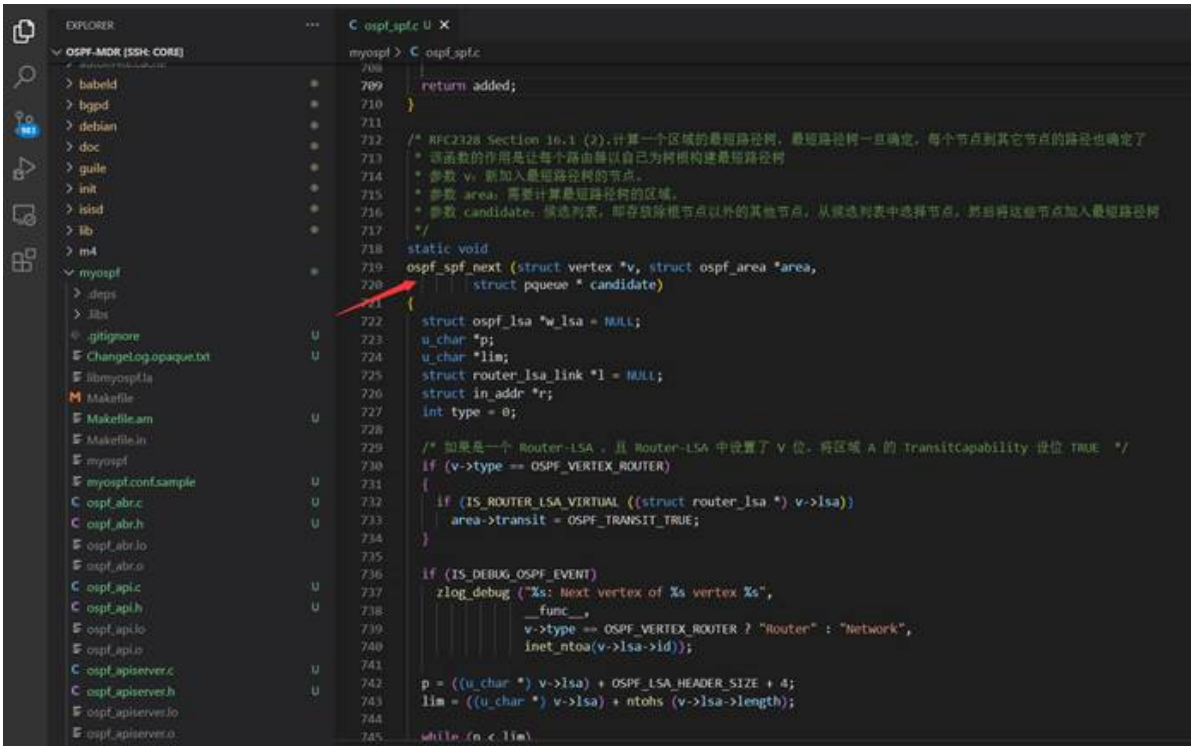




连接到服务器后，打开下图所示文件夹



更改ospf-mdr/myospf/ospf_spf.c文件中的ospf_spf_next函数



`ospf_spf_next (struct vertex *v, struct ospf_area *area, struct pqueue * candidate)`

函数说明：

- 该函数的作用是让每个路由器以自己为树根构建最短路径树
- 参数 v：新加入最短路径树的节点。
- 参数 area：需要计算最短路径树的区域。
- 参数 candidate：候选列表，即存放除根节点以外的其他节点，从候选列表中选择节点，然后将这些节点加入最短路径树

以下是OSPF最短路径树算法细节：

(1) 将新加入树的节点称为节点V。查看与节点 V 关联的 LSA（函数中的**while循环**）。这是在区域area的连接状态数据库中基于节点标识的查找。如果是一个 Router-LSA，且Router-LSA 中设置了V 位，将区域 A 的 TransitCapability 设为TRUE:

```
if (v->type == OSPF_VERTEX_ROUTER)
{
      if (IS_ROUTER_LSA_VIRTUAL ((struct router_lsa *) v->lsa))
           area->transit = OSPF_TRANSIT_TRUE;
}
```

在任何情况下， LSA 描述的每个连接都给出了到达邻接节点的距离值。对于所描述的每个连接 W（称为将节点 W 加入节点 V）：

1. 如果连接为stub网络， 检查节点 V 的 LSA 中下一个连接。到stub网络的连接在最短路径计算的第二步中考虑。
2. 否则，节点 W 为传输节点（ 路由器或传输网络 ），在区域 A 的连接状态数据库中查找节点 W 的 LSA （ Router-LSA 或 Network-LSA ）。如果不存在该 LSA， 或 LSA时限等于 MaxAge， 或不存在指回节点 V 的连接， 检查 LSA 中下一个连接 。 **（这部分不需要改）**
3. 如果节点 W 已经在最短路径树上， 检查 LSA 中下一个连接 。

   ```
   if (w_lsa->stat == LSA_SPF_IN_SPFTREE)
   {
       continue;
   }
   ```

   以下是计算最短路径树**关键代码**：
4. 计算从树根到节点 W 路径的连接状态距离值 distance。distance 是到达节点 V 最短路径的连接状态距离值（已经计算过），加上 节点 V、 W 之间的连接所宣告距离值。

   ```
   if (v->lsa->type == OSPF_ROUTER_LSA)
       distance = v->distance + ntohs (l->m[0].metric);
   else /* v is not a Router-LSA */
       distance = v->distance;
   ```

   如果 distance： 大于候选列表中节点 W 已经有的值， 检查下一个连接 ：

   ```
   if (w->distance < distance)
   {
       continue;
   }
   ```

   如果 distance： 等于候选列表中节点 W 已经有的值，使用所宣告的连接计算下一跳：

   ```
   else if (w->distance == distance)
   {
       /* Found an equal-cost path to W.
       * Calculate nexthop of to W from V. */
       ospf_nexthop_calculation (area, v, w, l, distance);
   }
   ```

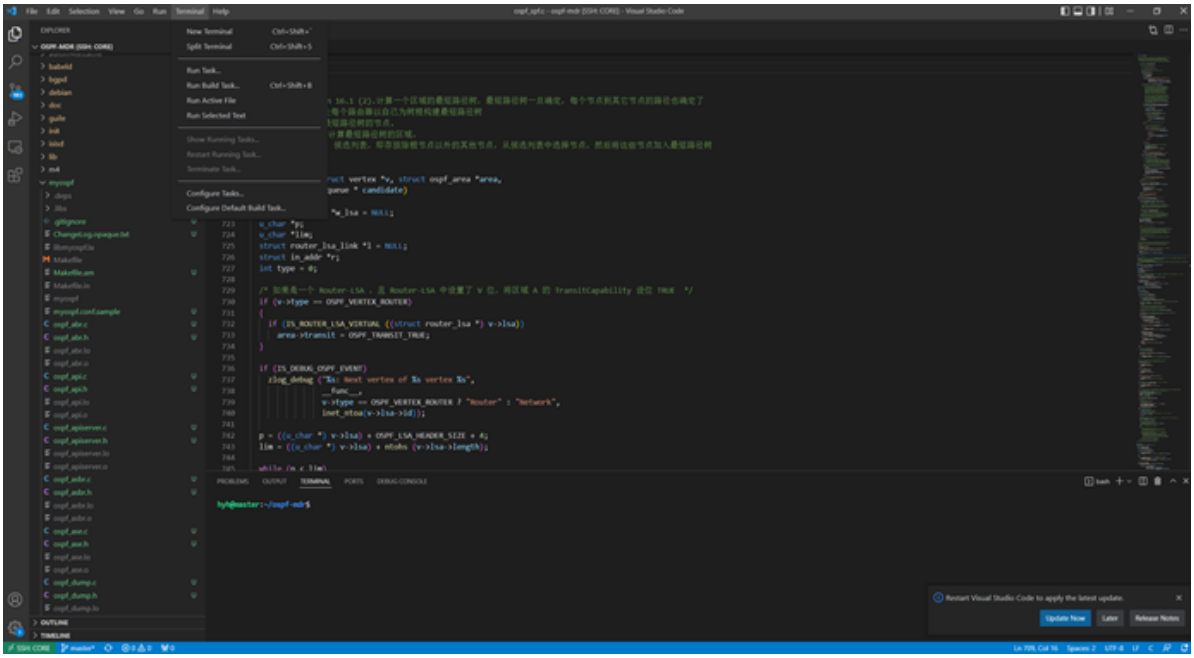
   计算的输入为目标（W）及其父节点（ V ），即`ospf_nexthop_calculation`函数。计算的结果应当加入候选列表中节点 W 的下一跳值。

   如果 distance： 小于候选列表中节点 W 已经有的值，或 者 W 不在候选列表中，将 W 加入 候选列表，并说明到达树根的距离为distance。同样使用所宣告的连接计算下一跳， 并以此设定 W 的下一跳值：

```
else
{
     /* Found a lower-cost path to W.
     * nexthop_calculation is conditional, if it finds
     * valid nexthop it will call spf_add_parents, which
     * will flush the old parents
     */
     if (ospf_nexthop_calculation (area, v, w, l, distance))
          trickle_up (w_lsa->stat, candidate);
}
```

如果候选列表为空，最短路径树 （传输 节 点） 就被构建完成， 这一部分过程结束。

对zebra源码进行编译，打开终端



\6. 在下方终端里依次执行以下指令：

`(1) make`

`(2) sudo make install`